SST Flight Operations Manual

Top Speed: Classified

Gadgets by Small SSTSSTSSTSST

40 W. Littleton Blvd.; #210-211 • Littleton, CO 80120 • USA • (303) 791-6098 • Fax (303) 791-0253

Limited Warranty Disclaimer License Legalese concerning Merchantability and Fitness

I always thought it was ridiculous to say "DO NOT OPEN THIS PACKAGE UNTIL YOU HAVE READ THIS AGREEMENT" when you have obviously already opened the package or you wouldn't be reading this. (Sound familiar?)

However, our lawyers, who are doing their best to make us into respectable business people, insist on us using the correct "legal language and terms thereof". So:

- Gadgets by Small, Inc. warrents that the SST will reasonably and generally conform to the written documentation for up to 90 days. Except as stated above, ***there are NO OTHER WARRANTIES EXPRESSED OR IMPLIED***, including the warranties of merchantability and fitness for a particular purpose, which warranties are specifically disclaimed.

- If the SST fails, within the 90 day period, to work as represented, Gadgets by Small, Inc. will, if the product is returned to us in usable form, refund your money or replace the product, at our option.

Anyway, in plain English this basically means:

- Gadgets by Small, Inc. is not responsible for how you choose to use this product, or for what you do with it. Also, Gadgets by Small, Inc. is not liable should any of your equipment or software or data suddenly quit working for any reason.

- You acknowledge that the SST software and such are copyrighted by Gadgets by Small, Inc., and belongs to us, no matter what. (That's fancy for: Copyright 1992 Gadgets by Small, Inc., All Rights Reserved.)

Warrenty
Disclaimer
License
Legalese

- You can make a backup/archival copy of the SST Software for your own use, but you cannot make copies for your friends, or neighbors, or dog, or anything/anyone else.

- You agree not to disassemble, modify, tweak, zap, or mangle the SST software in any way.

- Gadgets by Small, Inc. is not responsible for anything that may happen to you, your family, friends, neighbors, cat, dog, or other living being as a result of installing or attempting to install this product.

- You acknowledge that you have to get your Mega socketed yourself. We do not install sockets.

- This product is warrentied for 80,000 machine cycles, or 90 days, whichever is longer (90 days). Gadgets by Small, Inc. cannot possibly warranty that the SST will work with everything you try, because we don't know what you'll try; we have made a reasonable effort to make it work as best we can.

- You agree that by opening and/or using this product, you will abide by the terms of this agreement. (Gotcha!)

Thank you!
You've made our lawyers very happy.
Now all they want is your signature...

Table of Contents

Limited Warranty Disclaimer License Legalese iii

List of Illuminations viii

An Introduction to Speed ix
- An Excursion With the Small Family ix
- Out for a "Spin" x
  - "W A R P D R I V E !!!" xii
- And the Sound... xii
- The SST xiv
  - The Design Remains the Same xiv
- About this Manual... xv
- The Beta Testers xvii
  - SST Beta Tester Honor Roll xvii
- So... "On with the Show!" xviii

Video Wars 1
- Everything You Wanted to Know about Video, but... 1
  - The Big Picture 2
  - The Dot 2
  - Varying The Dot: A Picture! 3
  - ...Than To Fade Away... 4

Revenge: The Hard Disk Saga 7

The SHIFTER Strikes Back 11
- Introducing Video Memory 11
  - Dave's Easy Byte Definition 13
- The Incredible Burden Of Video Memory 14
- Static and Dynamic RAM 15
- A Stroke Of Genius: Shifter, MMU, DMA, and DRAM Refresh 16

Big Time Video Fun 19

The Revenge of the 68030 21
- Enter the Accelerators 21
  - Cached Accelerators 22
- The SST: Engine Philosophy & Design 24
  - The 68030 CPU: Why the SST Goes Mach 3+ 25
  - Our Own RAM 25
  - Math Coprocessor 28
  - Expansion Connector 28
  - A Cache? 28
  - TT Compatibility 29

TOS 2.05 & 2.06 From Atari 29
520/1040 ST Compatibility 30
A Scarred Veteran 31

Hard Disks, Floppies, and the SST 33
SST Disk Compatibility: Required Reading 33
TOS and SST RAM 33
Hard Disks and SST RAM 34
F-117 And The Fabulous Furb 35
F117: The Mysterious Stealth Buffer 36
Hard Disk Drivers & SST RAM 37
Atari HDX 4.03 37
Atari HDX 5 or HDX 3 38
ICD 5.2 through 5.4.2 38
Supra 3.43 38
Hard Disk Software Compatibility List 38
Heisenberg's Uncertainty Principle 39

SST Engine Installation 45
Installation & Checkout Overview 45
Before Installing the SST 46
AUTOMMU.PRG 46
COLDBOOT.PRG 48
RESET-Proof RAMDISK? 49
Static Electricity 50
Static Electricity Precautions 51
Socketing the 68000 51
In Preparation 51
Dismantling Your Mega ST 52
Remove the BLITTER Modification 53
Remove the 68000 54
Install the New 64 pin Socket 54
Reassemble and Test Your Mega 55
Remove the TOS ROMs 56
Prepping the SST: Installing Those Expensive Options 56
Installing the SIMMs 56
The 68030 & 68882 Pressure Intensive Installation 57
Oscillators 58
Putting it all Together 58
About Your Power Supply 59
It's Show Time!: A Tragicomedy 61

Pre-Flight Checkout 65
Power-up Video Check 65
Troubleshooting: The Warp Drive Isn't Operational 65
Checking the SST Board 66
Check the ST Without the SST 66
Retest the ST with the SST 67
Starting Up The SST: Warp Drive is Operative 68
MAKEDSK.PRG 68
Getting your Hard Disk Up and Running 68
SST fastRAM Test Flights 69
Boink Test 71
Reassemble Your Mega 73
Jenny and the Snowman 75

Test Flights 81
Wait States 81

Oscillator and CPU Speed 83
SST RAM Speed 83
Burst Mode Wait States 84
Big Time Really Super Neat Thing to Know 85
RAMnbmm.PRG Specifications 85
Tuning Burst Mode Wait States 86
Tuning Normal Wait States 88
Top Speed: Unknown 89
SST Flexibility 90
Eric Does Nikola 91

Take Off 95
Flying with Your SST Day to Day 95
The Cache 95
Saving The CACHE Status 96
SST fastRAM and ST RAM 96
ST RAM, No Cache (Low Speed) 98
ST RAM, with Cache (Medium Speed) 98
SST fastRAM, Cache On, Burst Mode (High Speed) 98
Compatibility: ST RAM, Cache, SST fastRAM 99
PRGFLAGS 99
Clear Memory Flag 99
ST or SST RAM Flag 100
ST or SST RAM Memory Request Flag 100
Changing the PRGFLAGS Settings 100
What Definitely Won't Work In SST RAM 102
Program Failure TechnoTalk 103
Other Day to Day Things 104
When To Initialize SST RAM 104
MEMSTAT 104
Go Have Fun! 105

Using the Spectre with the SST 107
Which Version of Spectre Do You Have? 107
Beta Testing 108
68030 Support in Spectre 3.0 109
It Is NOT a Bug! 110
Spectre 3.0 and the SST 110
Maximum Oscillator Speed 110
Floppy Disks 111
GCR Cartridge Timing and the SST 111
Floppy Disk Problems 111
The Future of the SST 113
Gadgets SST Support 115
SST Hardware Options 117
68030 Microprocessor 117
Coprocessor (Floating Point Unit) 117
SIMMs 117
Oscillators 118
Where to Order the Parts 118
George's Hard Disk Fix 119
Modifying the GCR for Warp Drive 121
SST Installation & Service 123
SST Expansion Connector Specifications 125
SST Cockpit Data Structures 129
Cockpit Quick Reference 130
Revision 1.0 SST MMU Data Structure 131
The Index 135

List of Illuminations

Illumination 1 - The Dot Computer 2
Illumination 2 - One Scan Line 2
Illumination 3 - Horizontal Blank 3
Illumination 4 - Vertical Blank 3
Illumination 5 - The Picture on the Monitor 11
Illumination 6 - Changing the Picture on the Monitor, by changing Video RAM 12
Illumination 7 - One Megabyte SIMM 15
Illumination 8 - Sharing RAM between CPU and Video 17
Illumination 9 - A Faster CPU can't get to Memory Faster 22
Illumination 10 - A Simple Loop 23
Illumination 11 - If the next Instruction is not in Cache, then Load it into Both 23
Illumination 12 - Load the next Instruction from Cache, not Memory 24
Illumination 13 - SST RAM; A Four Lane Super Highway 27
Illumination 14 - ST RAM Buffer for Hard Disk 35
Illumination 15 - The Mega ST Circuit Board 52
Illumination 16 - Pin 1 Orientation 55
Illumination 17 - The SST with 4 meg of SIMMs in Alternating Sockets 57
Illumination 18 - 68030 & 68882 "Fish Tails" 58
Illumination 19 - Oscillator Packages 58
Illumination 20 - Quick & Dirty Fix 66
Illumination 21 - The Mega ST Chip Locations 67
Illumination 22 - A Normal Wait State 82
Illumination 23 - A Burst Mode Wait State 84
Illumination 24 - An ST Timing Loop 97
Illumination 25 - The Same Loop on the SST 97
Illumination 26 - An Intelligent Loop 97
Illumination 27 - PRGFLAGS Display 101
Illumination 28 - Location of U2 and DMA Connector 119
Illumination 29 - U2 Changes 119
Illumination 30 - Pin 12 Under the DMA Connector 120
Illumination 31 - DMA Connector Area 120
Illumination 32 - GCR Warp Drive Fix 121
Illumination 33 - Underside of U10 122
Illumination 34 - The SST Expansion Connector 125

An Introduction to Speed

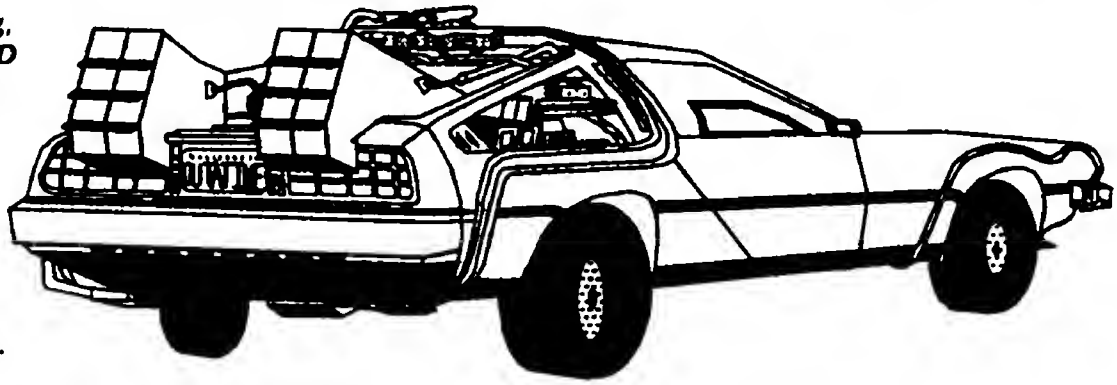
An Excursion With the Small Family

Open the long, long door, painted the cheerful red of the entire car (with the exception of the shiny black of the hood scoop). Climb into the driver's seat; your feet take about a mile to reach the pedals. Pull the door shut. The door was built in 1970; it seems heavy compared to car doors made these days. (It is heavier; it's built entirely of steel instead of polymers and plastic.)

The three kids clamber into the back, which really is a miracle; I can barely get myself into the smaller back seats, but the kids can seemingly stretch and squeeze through any obstacle. I guess back in 1970 the designers thought that back seats were really only decorative, especially in a "Super Sport" 396 Camaro.

Sandy, my wife, climbs in the passenger side seat, laboriously pulls her door shut, muttering about how difficult the car is to get into. Personally, I must admit to noting how well her legs go with the car's long frame, but I never tell her that. And I don't ever tell her that it is part of the American Dream to drive a hot sports car with a beautiful woman in the passenger seat; her ego, after all... Besides, Sandy manages the business that made getting my dream car possible! She turns, says, "Everyone buckled in?"

By now, the kids, familiar with the drill, are buckling away; Eric, who's practically 10 and the oldest, is helping Jamie, who is 3 1/2 and the youngest. Jennifer (8) somehow always seems to find the seat behind me and Jamie sits in the middle.



I know that it isn't a picture of a Camaro; but it is the "time machine" from one of Jamie's favorite movie(s): Back to the Future I, II,& III.

Jamie keeps saying, "Red car! *Red Car! RED Car!*" Jamie is clearly winding up towards nothing less than a thermonuclear explosion of excitement, grinning, bouncing, trying to see ever more out the windows.

In fact, when we were going through a book designed to teach colors, I pointed to a picture of a banana, Jamie said, "Yellow"; I pointed to a leaf, Jamie said "Green"; and I pointed to a picture of an apple. Jamie beamed at me, and confidently said, "Red car!"

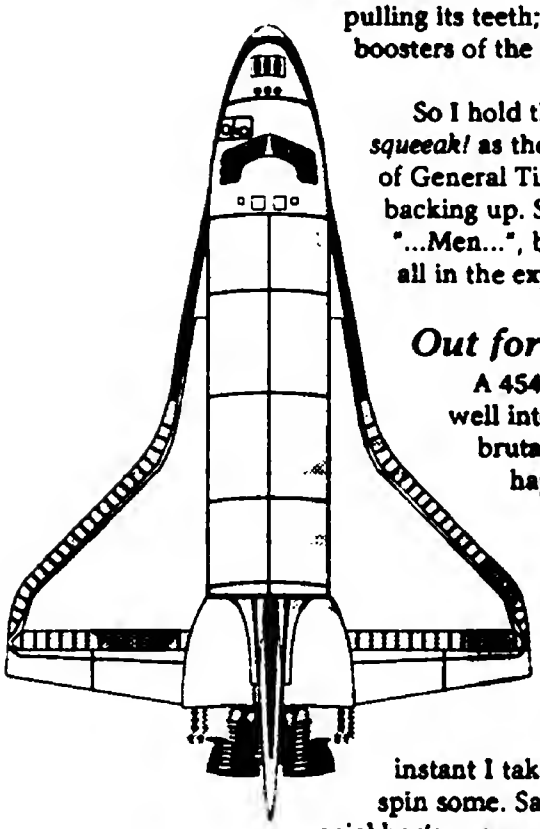
I buckle in the racing harness (sure is scratchy on my neck) and twist the key.

The starter grinds and moans. It's pushing against high compression, 10.25:1, as each piston squeezes the fuel/air mix to a fraction of its former size, then "goes over the top" and releases it. Squeezing the fuel/air mix that very tight brings about high efficiency, better mileage, and more power. Someday I'll have to invest in a racing starter; this engine is too much for the present "grinder".

Oh, I forgot to tell you. When we bought the car, the engine in it *wasn't* the 1970 fire-breathing 375 horsepower 396 cubic-inch monster; it was an engine from two years later, *after* the smog regulations kicked in. Any car nut ("gearhead", as Sandy calls 'em) will tell you that 1970 was the last year they made super performance engines; for instance, in 1971, all the compression ratios dropped to 9:1 or less, which was nice for nitrogen oxide emissions, but *really hurt* in the horsepower and torque departments. They even feed back part of the exhaust into the intake ("EGR")! Man, having a 1972 smog engine in a 1970 "SS-396" Camaro was to me an act of complete *heresy*. I mean, heck, in *these* sterile days, 8.5:1 is a typical compression ratio, and little computers run the engines to try and get mileage without too much smog, and compromise all the time; I know, because some hackers sell "performance chips" that you can plug into your car to wake it up...

Ah, a sputter from the engine. One of the cylinders fires and *twists* the engine through many revolutions *by itself*; the starter suddenly gets real useless. Pedal to the *floor*, hold it, and back, and the engine roars into life, *shaking* the car as it begins, then evening out as all eight cylinders start running with the smoothness that the 454 engine has always been known for. It really does *shake* the car; the engine twists so violently in the motor mounts that our body-work guru has an appointment with the Camaro to add hefty frame stiffeners, *lest the paint on the car develop cracks from frame twisting!*

The garage resonates wickedly with the engine, and I know better than to hang around; I've rattled things off of shelves with the sound of the Camaro! The amazing thing is that we went through great efforts to quiet the car, and largely succeeded without pulling its teeth; without mufflers, that 454 is as gut-frightening as the solid rocket boosters of the Shuttle.



So I hold the brake *tight* and put the transmission into "R" and hear the sudden *squeeak!* as the rear tires do half a spin on the garage floor, despite the best efforts of General Tires to keep traction. (The floor has many, many skid marks on it from backing up. Sandy looks at them and sighs; I believe the word she uses is "...Men...", but all the meaning of how she applies it is lost if I tell it that way. It's all in the expression and tone of voice.)

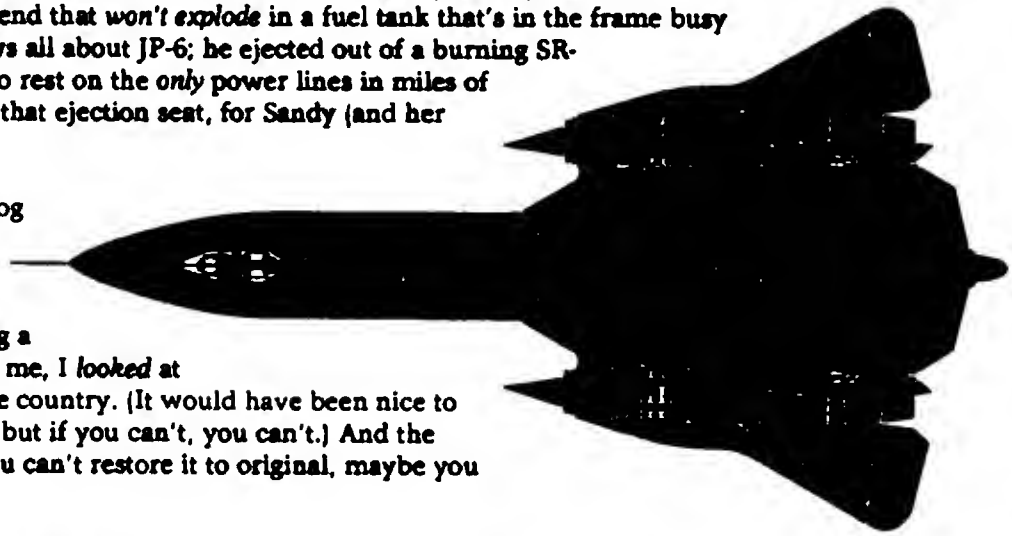
Out for a "Spin"

A 454 (8 liter) engine spinning at 2500 RPM (fast idle, on cold startup) is well into its "torque curve", where it is *well* on its way to supplying sheer, brutal *twist*. Not many people know it, but torque is where everything happens; that twist (torque) is what twists your tires and makes the car move. (Horsepower is more an intellectual measurement of the ability of the engine to accelerate mass over a period of time.) So when I put the Camaro into gear, when it's cold and thus fast-idling, I can't blame the brakes for not holding the rear tires down entirely. There is just too much engine in there and it's much too enthusiastic about waking up.

We begin to back out of the driveway, in fits and starts, for the instant I take my foot a little off the brake, the rear tires squeeeak! shrilly and spin some. Sandy has a forced, fixed smile on her face as she contemplates our neighbor's peace of mind when the resonant roar of the Camaro rattles 'em.

If anyone would know, it would be Sandy. She grew up at Edwards Air Force Base, in the California desert where the Space Shuttle lands, during an era where the Air Force was

testing darn near anything that would make a sonic boom, including the engines for the Saturn 5, *the most powerful machine ever built*. Sandy knows all about shaking houses and sonic booms! Her Dad Gary flew the world's fastest plane, the SR-71A, which sports twin wicked-powerful Pratt & Whitney turbofan engines, the world's neatest looking shape, and is made of a titanium alloy, because nothing else would hold together when the SR-71 hit Mach 3+. The titanium *glows red* at that speed; the fuel, JP-6, is *only used by the SR-71*, out of all planes, and is a special blend that *won't explode* in a fuel tank that's in the frame busy glowing red. Sandy's Dad knows all about JP-6; he ejected out of a burning SR-71 that went down (and came to rest on the *only* power lines in miles of vacant desert). I am grateful to that ejection seat, for Sandy (and her Dad)!



The SR-71A Blackbird (for those of you who missed the picture on the front cover).

So, I got rid of the 1972 smog engine, of course, and went to a Chevy dealer to find the original 396/375 HP motor. It was nowhere to be found, being a 21 year old engine, and believe me, I *looked* at all the Chevy parts places in the country. (It would have been nice to restore the car to original spec, but if you can't, you can't.) And the thought trickled into me... if you can't restore it to original, maybe you should restore it to *better*...

And then I noticed a sign at the dealer, Burt Chevrolet: **CloseOut Sale: LS-6 454's.**

Let me explain.

Everyone has an "LS-6 454" dream somewhere in them. It takes many forms. The person you **dream of for a mate. The perfect** house. The most completely cool computer. **The ultimate song a musician strives for.** The painting that the artist can't quite get on **canvas. The *dream*... I had read about the LS-6** *for years* in "gearhead" magazines, but ***always knew* it would be beyond me. But it** was one of my greatest dreams; my most-**wanted engine ever.**

The LS-6 is the most powerful street engine *ever made* by Chevrolet. It's a "Big Block", a design that debuted at Daytona in 1963, and which lapped the competition (and set new track records) just during qualifying! The LS-6 is "rated" at 450 HP, but everyone chuckles at that figure; someone made it up for insurance companies, so that buyers could get insurance for one of these fire-breathing Camaros (or Corvettes). These engines commonly show *550 HP* on a dynamometer. (The LS-6 has three close brothers; the 454 LS-7, and the legendary 427 L-88 and ZL-1's. Unfortunately, all of these engines are completely *"competitive off-road racing only"*; they burn aviation gas of 102 octane that you can't buy anymore since the supercars faded away, and they are not designed for minor things like "idling", nor mufflers, since these are race-only engines.). But the LS-6 had 10.25:1 compression, which combined with Denver's high altitude (5,280 feet, or about 1700 meters), enables it to barely get by, with super unleaded gas and some David tuning work, without pinging or flat melting pistons from premature detonation. (I doubt I will ever drive the Camaro to sea level, but if you see a red Camaro filling up with 105 Octane at an airport, stop and say hi, ok?).

Anyway, this makes the LS-6 the most powerful engine I could get that I could buy gasoline for, and run on streets.

We had a 'Vette shop install the LS-6, because they are used to all the touches the super motors require, from extra-large carbs to dual fuel feed lines to free-flow exhausts. And it turned out to be a good thing we did this; the old "smog motor" had a cracked and loose "torque converter-flywheel" housing, and it was just waiting for a bump to *fall into* madly spinning (100 revolutions per second) parts and wreak incredible havoc.

Really, the LS-6 is amazingly friendly, as long as you keep your foot off the gas pedal, or perhaps use the finest of caresses only. We "idle" down the street at 30 MPH. (At this point, I've only touched the gas pedal to start this thing!) The brakes get a workout at the stop signs. The engine warms up quickly, a property of high compression engines, which are more efficient. (Smog? Am I polluting the air with this nuclear engine? In reality, if an engine is cared for and in-tune, its smog emissions are *very low*, well within even California's restrictive regulations. This engine was designed to be efficient, and if you blow unburned gas out the tailpipe or other stuff that means incomplete burning and smog, the engine is not efficient! Really, it's the people that don't take care of their cars that really need "smog equipment". Let me tell you, the LS-6 in the Camaro is well, well tuned; that's where I learned its vibration and noise could rattle things off garage shelves.)

Finally, we come to the Interstate Freeway, (I am not going to say which one!!), but it's barely (if any) used at this time of day. The engine is warmed up enough; you *do not* push a cold super motor, or you end up with a handful of twisted metal pieces through the sides of the car. Anyway, I pull off to the side of the road at the start of the entrance ramp; at this signal, everyone turns to Jamie, the three year old, who *basks* in the glow of attention.

"Say it!", Jennifer (8) urges. "Go, Jamie!", Eric (10) says. But Jamie is patient, bides his time (someday he'll be a great poker player). The suspense builds; he knows He Is In Control, a *heady* thing for a little person just three years old with 500 HP!

And then when we've waited Long Enough, he shouts,

"W A R P D R I V E !!!"

We all echo him... and everyone braces themselves.

I let the brake off, *ease* into the gas. (If I punched it straight out, we'd start doing "circular doughnuts" right there and smoke 20,000 miles of wear right off the tires. I *know*, the hard way.) Despite my gas-pedal efforts, there are suddenly two dark lines on the pavement a hundred feet long, and I have had to work a bit to keep the car straight. (The kids think the squealing tire sound is great, I'll tell you!) After we hit fifty, in a couple (maybe three?) seconds, I floor the gas pedal and the carb, where air and fuel mix, says "Hey, it's rock and roll time!"

Inside the carb there are "barrels" where air rushes through and mixes with gasoline in a fine spray; normally, the Camaro operates on just two of them, the "primaries". But when floored, another two barrels ("secondaries") open up, and begin *enthusiastically* dumping gasoline and air into the engine. In fact, this *particular* carb has 750 CFM (Cubic Feet of air per Minute, 27 cubic meters/sec) *ROARING* through it; the loudest noise the whole car makes is from the carb *intake*, through the hood scoop, not the exhaust noise (which we quieted). The LS-6 takes all this potential energy of gas and air being fed to it, winds up, and *twists* the rear wheels. The acceleration stiffens and I hang on to the steering wheel tightly; my arms pull towards me from the acceleration "G" force...

And the Sound...

And the sound... I must tell you the sound alone is the best thing about it, the *reason* I do it. It's the sound of a hot engine winding up, mechanical racing cam slamming the valves shut and lending a clipped note to the sound; the same sound you hear from a rock guitar winding up an amplifier past its limits. It gives me goose pimples. In Spectre 128™, I spent 125,000 bytes of the release disks on a digitized sound file of just this sound, because to me it was the sound of "pushing the envelope"; it was our "dedication page" I gave

directions to in the manual. The group "Boston" has this sound down pat, and Tom Scholz from that group now sells the "Rockman™" to let you get that sound; I have one.

The engine claws and screams its way up to 6,500 RPM, where the cautionary yellow line is (red is 7,000 RPM), and I shift into second out of first. Given that the speed laws in this country max out at 65 MPH (about 100 KPH), *I won't say* that we exceed that limit... but I *will* tell you that I just shifted into second gear, out of three (automatic transmission; clutches do not live long behind an LS-6, so I went automatic with an especially strong "built" transmission).

Second gear is sort of a relief. You're no longer *shoved* back into your chair and your cheeks don't feel funny from the "G" (1.0 G = Earth Gravity) forces on you. You simply continue to watch the speedometer wind clockwise at ridiculous speed. This is when they still put 150 MPH speedometers in cars, and *meant* it. And, this acceleration is happening at a time when wind resistance should be slowing things down! (Ah, but the hood scoop on the Camaro just *lives* for wind; it pulls in the wind, and feeds the resulting pressurized air into the carb, force feeding it, with cooler/denser air, for *even more* power. See, you *can* win sometimes!)

Now the fine suspension of the Camaro comes into play. We're still accelerating heavily, but we're not unstable; the Camaro sorta hunkers down and grabs the road. The suspension has the incredible ability to corner at *more* than 1.0 G, which is the theoretical limit. And the spoilers direct the rushing air to *pushing* the car *down*, to help stabilize it.

I've never run the Camaro on a formal race track (just you wait... Bandimere Speedway, Summer '92, here I come!), where a standard acceleration run of 1/4 mile is timed, and your speed at the end of the distance is logged. But it is very few seconds indeed to come off the stoplight, go up the on-ramp, to get to where I am *slowing down* to merge with traffic (if any; I usually don't do one of these Banzai's with traffic around).

Somewhere in second gear, I very reluctantly let off the gas some, and the secondary throttle valves slam shut, and the screaming sound of the engine that I so love disappears. *I do this because the Camaro is approaching the design limits of its tires.* I suppose if I bought Pirelli tires, which have a higher speed rating, I could *easily* hit the RPM limit/rear-axle-gear-ratio/tire-size calculated top speed of 170 MPH. There's no question the engine is there to do it, the horsepower to push against the massive wind resistance at that kind of speed.

But I *went* 150 MPH in high school at age 17 in a souped-up Firebird... and I'll tell you, there's so little room for error at those speeds, where the white lines on the road blur together into a single line, where you must keep your eyes miles ahead, watching for curves, cars, stoplights (ouch!) or the local speed-limit people. (The guy driving ran into them, and has the traffic ticket he got for it framed, hanging on his wall; 140 MPH clocked, in a 55 MPH zone. He got in considerable trouble, as you might imagine.) That speed is exhilirating, but also deeply terrifying.

Into third gear, regular old "Drive", and we're pretty much done with the exciting part. Oh, sometimes if a car gets irritating, we "hunt it" and pass it by. (I finally explained to Sandy how to drive the Camaro by telling her to hunt down other cars instead of driving *with* them; after that, she did fine.) It isn't much challenge, and I admit to the immaturity of effortlessly going by people in modern day "fast sports cars". (And every now and then, I run into someone else with an old super-car like mine, and of such things I will *only* say that we both smile and wave to each other when we part, miles later.)

But I'd like to tell you... for the very long seconds while we're winding up in first gear, *the sound* living in my ears with the beating of my heart, the utter power of Chevrolet's best pushing that car...

... my soul flies, and I am free.

The SST

That is speed, and *that* is why we built the Camaro, *and* the SST accelerator board for your machine. We like to do this. (Of our other products, MegaTalk gives you the fastest serial and SCSI you can get, and Spectre is *well known* to whomp on Macs in drag races.) The SST brings your Mega ST into the 90's, in many ways.

After you plug in your SST, you'll be blown away for a few days by the speed of "everyday" operations. Then, you'll get used to it, and get to see your friends getting lustful after seeing the SST. Then, you'll be so used to this *level of performance* that it'll become your *standard*. And it's fun.

Other Editor: That's why I love to take people for rides in my Camaro, I admit...

We have been testing the SST with our loyal Beta Testers for many months, and have seen the same thing happen over and over.

It grows on you, this SST does. Little things. Windows don't "open" anymore, with animation "grow-boxes" showing them opening, like you can see on an ST. They simply snap, Closed to Open, in one eyeblink. Doing software development, I crunch pretty large files In, oh, three seconds. (It used to take thirty.) The file selector box scrolls by so fast I just pick a spot in the scroll bar I think is close to the file, hold down the mouse, and in no time we're there. (Before, it took too long to do that.) Same thing with scrolling in text files; you will often find yourself zooming straight to the beginning or end without meaning to, until your habits change. And the "Boink" program, with its bouncing ball, changes from a slow "bounce...; pause, bounce...; pause, bounce...; pause" to something sounding like a machine gun!

I guarantee you will *really* appreciate the SST only after the first time you go back to an ST (I *know*.) All the little things you've gotten used to disappear, and that machine that seemed so fast when you first saw it... well, it's dragging its feet. You'll *really* appreciate getting back to the SST, believe me!

In a way, if you're lucky, you'll be going back and forth from an ST to an SST machine. That'll keep your feel for its speed fresh. I do the same thing. I drive a van to school sometimes to pick up the kids, and trying to get up speed to merge with traffic on an on-ramp, or cornering, reminds me real fast I'm not in the Camaro.

The Design Remains the Same

It shouldn't surprise you that the SST design follows the 454 Camaro's design in many analogous ways. Heck, there's only a few ways to get fire-breathing power out of anything, and engineers know them. It boils down to this: the amount of gas and air the engine can get, *per second*, into, burned, and through the engine, determines how much power it can output that second. Thus, to get high performance, you typically use a big engine (thus, more air and gas get pulled in naturally per engine spin) and you feed it as much gas as it wants, with a big, efficient carb, like a 4-barrel, and tune to ensure it burns all the fuel it can handle. Planes aren't any different (just faster!). The SR-71 uses huge Pratt & Whitney turbofans with air intake tuning, and the pilot could turn on afterburners which simply dump raw JP-6 gas into the engine for more performance. (More information is still classified.) The Camaro uses 454 cubic inches (8 litres) of engine, a 4-barrel carb, tuned exhaust, and twin fuel lines, and does it work!

So in the SST's case, we used a big, 454-style engine (Motorola 68030), a 4-barrel carb (brand new technology 4-byte-wide DRAM controller) for afterburners, added a supercharger to get enough fuel to the carb (like the Camaro's twin fuel feeds, called Burst Mode RAM circuits), and enough high octane gas to feed it: 8 Megabytes of memory, which adds to whatever memory is in your machine. (For instance, a Mega 4 can become a

kind of "Mega 12"). Designing the SST was fun for me personally, for we made the same decisions I made with the LS-6 Camaro; we went for maximum performance possible. It's not surprising we followed the engineering principles for high speed that people like Kelly Johnson (SR-71A Blackbird chief designer) used, that the Chevy Big Block design team used (many people, I wish I knew the engineers' names); they're time-tested and proven.

There's an old racing proverb: *"Speed costs money. How fast do you want to go?"* Computers with speeds rivalling that of SST cost beaucoup bucks! Yet with SST, we deliberately used parts in the design whose price has been hammered down by intense competition in the PC and Mac arenas, and managed to keep the price reasonable for the performance; we targeted "under a thousand dollars" at spec time, yet came in at $599 retail for the bare essentials!! Atari computers always were "Power Without The Price" ™; we like to think we've given you a product along those lines, that *you can afford*, compared to anything else.

I would like to share my dream, the SST, with you, to let you have a 1990's Mega ST. You can order the SST configured *your way*, at the speed or price, you can afford. Then you can upgrade.

You see, the ST is fundamentally a pretty good design - for *1984*, when it was laid out. However, it is lacking in a few things that the *1990's* require of a machine, namely *horsepower* and *memory*. The SST brings your ST into the 1990's in *very fashionable style*. (The third requirement, High Resolution Color Graphics, is also on the way, from Gadgets and other manufacturers; the fourth, Networking, is what our MegaTalk product is all about.)

Oh, yes, the SST does "warp drive"; Jamie would be proud (if he understood it). For instance, SST RAM can run a program that is 8 million "bytes" long in *around half a second* (that's 7 double-sided floppy disk's worth!). It would take an ST around 12 times longer to do that, but heck, the ST couldn't hold such a large program to begin with!

Believe me, if you're doing something that makes the computer work hard (desktop publishing is usually a biggie), you are going to really sit up and notice the SST! It still surprises me, and I often "drive" a Mac IIfx, one of Apple's fastest!

You know... sometimes I drive The Van to pick up the kids from school, and Jamie hopefully shouts "Warp Drive!!!". But nothing happens if I push on the gas pedal, and he is disappointed. The look on the face of a disappointed three year old is *very* expressive. Similarly, I have to drive 8 MHz ST's from time to time, and I know just *exactly* how Jamie feels! Sure, The Van gets there and home again alright and can hold the kids, too.

But I tell you...

... I would rather be driving my Camaro, and the SST. Or an SR-71.

About this Manual...

I have long believed computer manuals to be among the worst expressions of the literary art in history. I have read many and been bored beyond rational belief that *any* human being *wrote* that stuff! I can only imagine a machine, not a human, being patient enough to wade through the pap and extract the needed information. It's even worse when they "dumb-down" a manual and hide information from you (mutter, mutter!) because they believe you're essentially brainless, and heck, you won't read it anyway.

My belief is that the SST is for human beings, and the manual should be, too. Not for computer wizards, just for people. After all, the SST is intended for people who use their ST and who want to bring it into the 1990's.

Believe me. Bringing up the Mac operating system on the ST required me to be very intimately knowledgeable about *both* computers (makes sense, right?), and I've read some positively dreadful manuals. We could probably solve prison overcrowding by having convicts sentenced to read computer manuals; no one would DARE commit a crime if they had to read the books of developer documentation for "&#£¢∞§¶•ª° !!™, Version 3.x" as punishment.

Editor: Our lawyers insisted we delete the name of that program. Sorry, Dave. I think its manuals are the pits, too. But the program isn't that great, either. Maybe they are charging so much just for all the paper in the manuals!

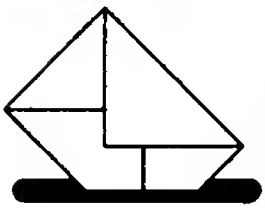
One thing that helps human beings is to break up long manuals with short anecdotes and vignettes that have *human interest*. It keeps the manual from being one long techno-dweeb talk session, and it also keeps the link between the author and the reader (that's me and you!) alive and kicking.

So you will find Interludes between chapters and in other areas of this manual. They are there for your enjoyment, *because it means something to me if what I wrote was enjoyable for you to read.*

We did this same thing in the Spectre 128™ and Spectre GCR™ manuals, and found them among the most-read manuals of any computer product anywhere - and swarms of customers telling us the manuals were excellent and *readable*. Spectre pushed the envelope in more than one way.

Finally, the Interludes are a way of showing you something about us. Gadgets by Small, Inc., *is a family business*, and already, in the Spectre GCR Interludes (and this Introduction), you've met the whole gang, and gotten some sort of idea what we're like. (Yes, we have outside contractors that program and do other stuff for us, but fundamentally, it is a family business). We are staking our *livelihood* - you know, house payments, all the same bills you have - on our reputation and on our products; that someone will find them worthwhile enough to pay some pretty hard-earned cash for them. We think you deserve to know a little about who you're dealing with, and many of our customers have become good friends over the years; we like "doing business" that way.

Suggestion Box



I don't know why, but companies after a certain size become faceless; in-house politics strangle new, good ideas, and they all look alike. I have worked for outfits like that, and I have left them. I *much* prefer talking to customers, be it by phone, FAX, or online networks, and keeping things on a much more personal level. Again, it has many positive side effects - for instance, I often get very, very good suggestions from users that I can implement into our products - but at the basic level, I have only one career to span this 33rd year of my life, and I've found I like to do things this way.

Unlike faceless companies, we admit to goofs, and publish a newsletter, the Gadgets News-Herald-Update-Enquirer (plus all sorts of online work to keep people who want to know quickly up to date), to let users know about goofs, our fixes for them, plus new things we are doing. Faceless companies feel that admitting to even blatantly obvious goofs tarnishes the company's reputation for being "perfect" or something, and just TRY! to get ahold of a programmer to talk about bugs in the program. (I remember calling Digital Research, making it to a programmer, and having the astonished programmer ask me, in detail, just how I had gotten through to him... "That's never supposed to happen!")

I might be wrong, but I think customers prefer our style. I prefer being able to be honest about late shipdates or bugs rather than stiff upper lipping it. I believe the online networks and user groups and magazines prove that people are hungry for *honest, accurate, and fair information*, from the horse's mouth maybe even! I know that people seem astonished that they get to talk to me on the phone, and make suggestions; I can tell they've been talking to BigShot Computers, Inc.

In return, I have seen customers do things that amaze me, that show their loyalty. We typically find notices that someone (Bob Brodie of Atari, in particular) has demonstrated the Mac emulator to a 500+ member Mac user group (talk about a hostile, "tough" audience! That would scare me to death!) I've seen letters written to Mac magazines complaining about errors in their reviews of the Spectre products. And most heartwarming to me are the comments people put on their warranty cards about the product; I can only hope that SST pans out as well as Spectre GCR did. When I feel down, I go read some warranty cards. The people didn't *have* to write anything, but they did.

The Beta Testers

We have also had the help of some fine people, the Beta Testers. The name is a computer-geek-greekbased-word; a product is initially made, "Alpha Tested" for horrible bugs (which is generally done "in house"), then "Beta Tested" for bugs no one knows about, by people that use the product for awhile (sometimes quite awhile) before general release. Sometimes there are also "Gamma Tests" for a final checkover.

Other Editor: Greekbased: Alpha, Beta, Gamma...(a pun)

Editor: We ought to use the "Alpha, Bravo, Charlie, Delta, Echo, Foxtrot..." code. Then we could call them "Bravo Testers", since they're so enthusiastic!

The Beta Testers have all been with us a long time, several years, and because of them, the SST works a heck of a lot better for you than it would have otherwise. These guys have literally had hard disks lose all their data on them to *insidious* subtle bugs, and other hideously charming things like that, during the SST Beta Test. They have also been patient beyond belief with me (I know *I* am not a happy camper when my hard disk dies). They tested the SST day in and day out, reported bugs, tested fixes, reported bugs in the fixes ("sigh"), tested fixes to fixes, and so forth. They're all heroes, at least in our eyes.

We couldn't have done it without them.

They deserve your thanks; if you see them on the GEnie network, drop them a line and say something, ok?

SST Beta Tester Honor Roll

Mark Booth	**(STACE)**
Jeff Greenblatt	**(JEFF.G)**
Bruce Rogovin	**(B.ROGOVIN)**
Norm Walker	**(SACTESTER)**

For you long time Gadgets customers, Doug Wheeler (DOUG.W) is now working at ICD these days, and can't Beta Test anymore because of contract conflicts. We try to keep in touch anyway, and Doug has also been very helpful in the SST project, so we'd like to extend our thanks to him, too.

Finally, I would ask that you keep in mind that we are a family business; the economy in general isn't that great as of this writing; and sometimes we can get overwhelmed with work and be late getting back to you. We are not BigShot computers... and we are proud of that. We also have kids in school swapping diseases with other kids and bringing them home for us; I just missed a big Atari show because I was sharing a cold with one of my kids! We *do* make goofs. I appreciate the patience many of you have extended to us in getting SST's shipped out; in return, we worked many a long night to get them to you faster. (*It is 2:12 A.M. as I write this!*)

Editor: And 3:20 A.M. as I edit this, and I have to get the kids to school in the morning! Of course, Dave is in his office adding the finishing touches to the SST software, so I'm not the only

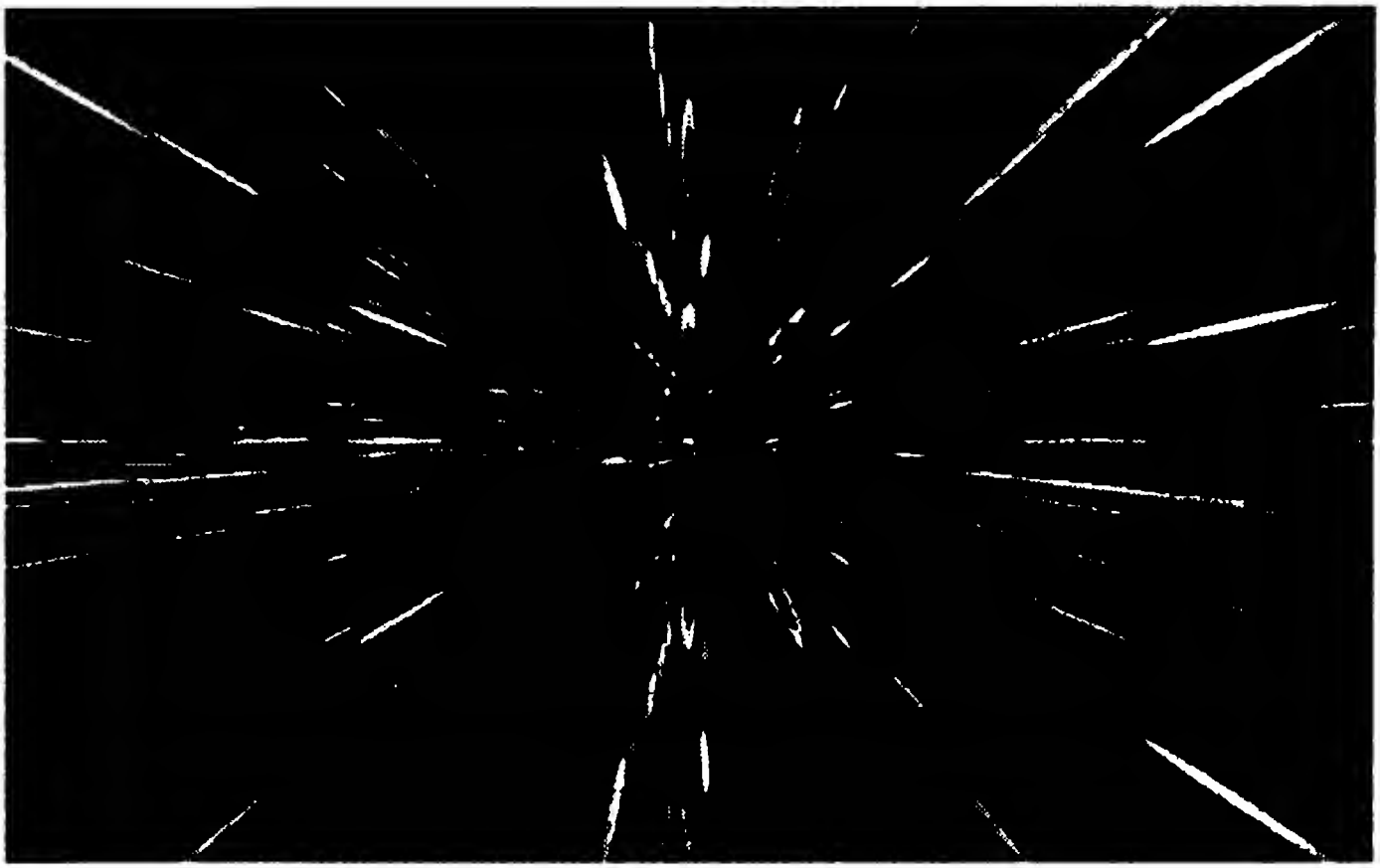
person up at this hour in the Small household.

Other Editor: Our dog Fang is awake, too.

So... "On with the Show!"

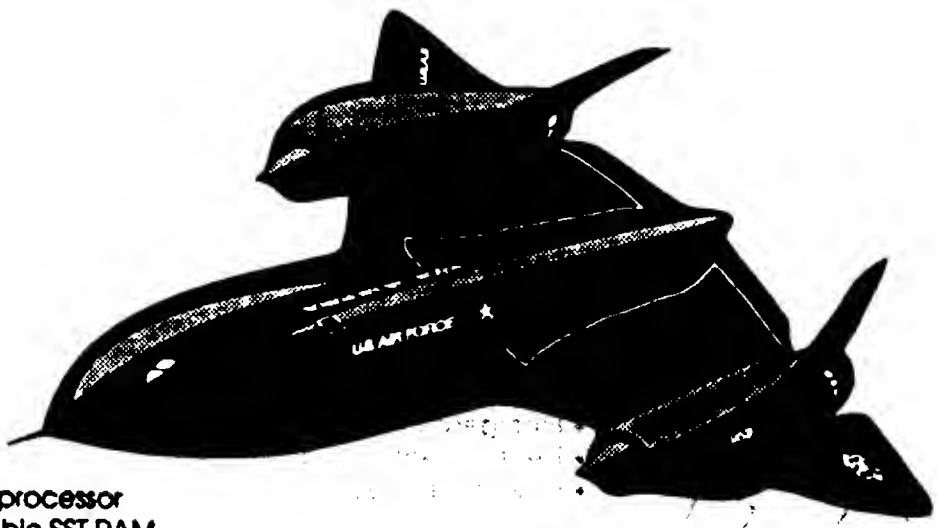
And now, on stage, live, for the first time:

The Gadgets by Small Incredibly Super-Fast Splendiferous Totally Amazing 68030 SST!



SST Flight Operations Manual

by David and Sandy Small

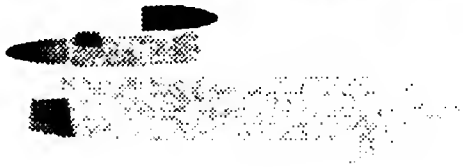


SST Capabilities:
33 MHz 68030 32-bit Processor
33 MHz 68882 Floating Point Coprocessor
8 Megabytes Burst Mode Capable SST RAM
Atari TOS 2.05 (or 2.06)
George Bus (Expansion Connector)
Software by David Small
Runs on JP-6 Jet Aviation Fuel (not really)

SST Flight Operations Manual

Written by: David M. Small (as usual)
Edited by: Sandy Small (as usual).

Extremely important: Our lawyers insist that you (right now) go read the Limited Warranty Disclaimer License Legalese Section concerning Merchantability and Fitness!



About the Cover:

The SR-71A holds a special place in the hearts of we at Gadgets by Small, Inc. For many years one of the test navigators of the SR-71A was Lt. Col. Gary I. Heidlebaugh (ret.), father of Sandy Heidlebaugh, who became Sandy Small, my wife.

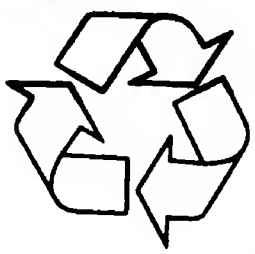
The Gadgets SST is hereby dedicated to:

- Gary Heidlebaugh, for being my Dad.
- Kelly Johnson, the primary designer of the Lockheed SR-71A Blackbird which, after a 20 year service life, still holds the world record for altitude and speed.
- All those people with the courage to stride boldly into the undiscovered country.

This manual contains the following (more or less!):

- Why It's So Fast
- Engine Installation
- Pre-Flight Checkout
- Engine Start-up
- Takeoff
- In-Flight Procedures
- Landing
- Shutdown
- Emergency Procedures
- Ejection Seat Operation
- How To Get Help In-Flight From Gadgets

This manual is printed on recycled paper.



A long, long time ago, in a basement far, far away, it was "difficult times" at Gadgets. The children were stomping on the ceiling, the phone was ringing off the hook, the fax machine decided to work overtime. Barb started her own business in Jamaica. Christmas was coming, and the SST Manual wasn't ready to duplicate... "sigh"... and then Atari released TOS 2.06...

Video Wars

Before any pilot takes up an SR-71 for the first time, they are thoroughly briefed on *why* and *how* to make it work up to it's full design potential: i.e., "how to get the most out of it". That way, they don't try to do things the plane wasn't designed for! You *need* the same kind of "briefing" for the SST; *you* just don't have to spend months in a simulator!

First, I'd like to tell you about the Atari ST's innards and why we have to take a particular approach towards accelerating the ST. Throughout this section I'll take particular pains to speak in English! **But you must know a little about how the ST works in order to understand how we accelerate it, where the 68030 and fastRAM fit in the Big Picture, and whatnot. *Only then are you going to hit Mach 3+ in your SST.***

Second, I'll tell you about the SST and its abilities, and how it's designed to work *with* the Atari ST to give you speed. When the ST was designed, back in 1984-1985, no one dreamed of accelerating it with a 68030, and it was really quite tricky for us to figure out a way to attach an accelerator. You get to see the results - but you should know it took 2 years and thousands of man hours to tie it all together.

As a bonus, during these chapters I'll be introducing you to some terms that you won't ever see *me* using in public (like "video contention"). The computer industry has spawned its own jargon that is incomprehensible to the general public (kind of like doctors). That's sad; computers are not that impossible to understand, but one of the big things getting in the way is all the jargon.

By the time you're through with this section, you'll *understand* some of that jargon, and never have to feel puzzled (or, honestly, intimidated) about it, ever again. Heck, that's worth reading a few pages, isn't it?

If you don't read this section, parts of the SST will always be a mystery to you, and ***you won't be able to "fly" the SST as fast as it can go.*** I promise, it's written in plain english, and has even passed review by my Mom, a known computerphobe!

To begin with, we need to tell you about the Atari ST, and why it is locked into its present speed. Then you'll understand why the SST is designed the way it is, and that'll tell you a lot about how and when to use the SST. Let's begin at a place that might surprise you: the screen of the ST!

Everything You Wanted to Know about Video, but...

Imagine turning on your ST. In a bit the "desktop" pops up, and the machine sits there showing it to you. Since nothing appears to be happening (nothing is moving), most people think the machine is not doing anything.

This is just illusion. In reality, the machine is working its heart out keeping that video image up there - and if anything goes wrong with that full-out effort, your machine crashes, from the screen going black to the well known "bombs" onscreen or a "lockup" where the machine just "freezes" on you.

What you are seeing on that monitor screen is something close to a miracle. 99% of the people in the world who are using computers have no idea where the display comes from (and there are some fun and amazing things to know about how that display stays there).

Editor: And if you already know all this video stuff, you can just skip directly to the Interlude.

The Big Picture

Let's start with a "monochrome" (black and white) display because that's easiest to explain.

The picture tube has a big glass front, that you see, and tapers down to a small back, sort of like a champagne or wine bottle. There are two parts to the display: phosphor, which you look at, and an electron "gun" (the "cork" end).

The part of the picture tube of the monitor TV that you can see is plain ol' glass with a weird material ("phosphor") painted on its inside. Phosphor has a strange property: If you shoot electrons at it, the screen glows brightly, *while the electrons are hitting it*. When the electrons cut off, so does the phosphor, and pretty fast - in about 1/30th of a second, but that number varies. If you'll think of radar displays you've seen on TV or movies, with a circular line sweeping around, and where a "dot" is "painted" onscreen and fades, well, that's a perfect example of a phosphor fading, and which needs to be repainted now and then.

There's also an "electron gun" at the back of the picture tube. As you would imagine, it fires electrons! It's in the skinny back part of the TV tube. The electron gun fires a very tightly focused beam at the screen, one "dot" size large, where a dot is the smallest dot you've ever seen on an ST screen.

Illumination 1 - The Dot Computer

The Dot

Now, let's create a computer picture (display) with this setup. We set up some things to allow you to aim the electron beam coming out the electron gun ("deflection plates"), and we point the beam at the upper left hand corner of the display. We kick on the beam, and wow!, you've got *exactly one* tiny glowing dot at the upper left hand corner of the screen. It's the size of the littlest dot you have ever seen on an ST display. That beam is tiny!

Well, that's *something*, true, but no one is going to buy it ("The Dot Computer", I can just see it - and so can you, in Illumination 1). So, let's tell the deflection plates to move that dot to the right. You see the dot "sweeping" across the picture tube in front of you, a little like how a radar display sweeps in a circle; eventually the dot reaches the right hand edge. As the dot goes by, the phosphor lights up, then it fades away as the dot moves on. We reach the right edge; we have just painted a "scan line", which is just one horizontal line. (See Illumination 2.)

Illumination 2 - One Scan Line

(These jargon terms - deflection plates, scan line - are pretty straightforward in English, aren't they?)

Now, let's turn off the electron gun; the dot disappears. Let's move the aiming point

Illumination 3 - Horizontal Blank

back to the left side, with the electron gun off (so we don't paint junk on the screen while it moves back), move the aim one row down, (oh, about 1/72 inch?), turn the gun back on, and sweep across again for another scan line. Now we have two scan lines, the second below the first. (Illumination 3.)

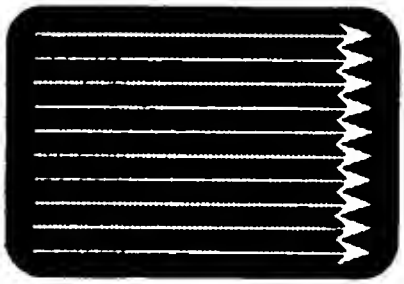
The time the beam is off, as the gun goes from right back to left invisibly, is called "Horizontal Blank", by the way. You can impress people that you're a technowhiz if you'll just use some of the words I'm telling you about at computer user group meetings.

Suggested Question: "Does your program use the Horizontal Blank Interrupt for its display?". I guarantee, you'll become an instant High Priest of Computerdom.

Other Editor: Don't laugh. consultants do this all the time for money. Hmmm. come to think of it. I'm a "consultant"!

So, we have swept the beam across the display, left to right, and returned it to the left one line down. Continue this for 400 lines! At the end of the 400th line, we're at the bottom right of the display. We've just done one "screen refresh"; in other words, we've "painted" the entire surface of the TV screen with the electron beam. Now remember, and this is important - *that screen doesn't stay painted*, it just briefly lights up as the electron beam forces the phosphor to glow.

To complete the process, we turn the gun off, point it to the upper left top of the screen where we started, and start all over! The process of returning to the top is called VBL (Vertical Blank). Programmers use VBL all the time to synchronize animation to the video screen. (Illumination 4.) It's another *great* buzzword to use.



Illumination 4 - Vertical Blank

On a monochrome monitor, it takes right around 1/70th of a second to do this entire process. This may not seem like much to you, but let's think about what just happened. Let's arbitrarily say the display is a foot wide (that's close enough). There's 400 "scan lines", so we just "painted" 400 feet with that electron beam in 1/70th of a second!

In one second, we paint 70 times; that's (400 x 70) or 28,000 feet, about five miles or 12 kilometers! And in just one hour of you using the ST, or 3600 seconds, you've painted 28,000 feet x 3600 seconds, or 100,800,000 feet - 19,000 miles (41,800 kilometers)!

Okay, back to that video display that we've painted once. (I told you some of this stuff was neat!)

Well, all we have to show for our efforts is a screen that briefly lights up, as every location on it is hit by the electron beam. It lights up for only a small fraction of a second. Blink, and you would miss it! Well, Woof! (That's Computer Hacker slang for, "Big Deal. Hmmmph.") We want information on the screen, not just a quick flash of white light! Why, I can get *that* just from banging my head against the wall!

Varying The Dot: A Picture!

To make a solid-white screen into something interesting, with black and white, we vary the *strength* of the electron beam. To get a "black" dot, we crank the beam down; to get a lit-up white dot, we turn it back up to full power. (In monochrome, we only have ON or OFF; with a color monitor, there are in between levels.) The people who designed the ST decided there would be 640 places to turn the beam on or off on each "scan line" going across; those places are called "pixels" (Picture Element). You've seen a pixel; it's an

individual dot on the computer screen, very small. It's the *smallest possible dot.*

Now if you look very closely at your monitor, you will see that *everything* on there is created out of these small dots! The "icons" are just patterns of dots; the "background grey" of the desktop is just a pattern of on-and-off dots, and so forth. If your monitor is a little out of focus and fuzzy, it'll be harder to see; try a "painting" program, draw a narrow diagonal line, and you will easily see the "stair steps" or "jaggies" as the line climbs from one scan line to another.

Now's where it starts getting absolutely incredible. As it turns out, monitors will handle sweeping the beam back and forth and down, then up, all by themselves, without a computer, something like a TV not tuned to any channel. They're pretty relentless about it. But folks, the information to turn the beam on and off, to actually "paint" a screen with information on it, has to come from somewhere – and that somewhere is the computer.

In 1/70th of a second, it has to tell the TV to turn the electron gun on or off 640 times per line for 400 lines, or 256,000 individual decisions, in just 1/70th of a second! That's moving *fast*. In terms of decisions per second, that's 256,000 x 70, or 17,920,000. *Seventeen million decisions per second!*

As my wife Sandy, the Editor, will tell you, getting two decisions per day out of me is doing pretty good...

Editor: Well, it's not quite that bad!

You've seen this many times and not known it; think of when you turn on your ST. You click on the computer, and the monitor is told to "start going"; it mindlessly sweeps that electron gun through its pattern, over and over, without the computer telling it to vary the beam's strength from "on". So you end up with a white screen! (In fact, if your ST ever fails to start up properly, you'll often be left looking at a boring white screen... I've seen it many times.) Then, when the ST knows what sort of picture it wants to display, and the video circuits start waking up and talking to the monitor, the monitor actually displays a picture (the desktop); that's the point when the electron gun starts spitting out varying levels of beam, to make the picture on the screen.

Now, I'm sure you're thrilled to know that your monitor has traced 19,000 miles in the last hour, but you're saying, "What does this have to do with the SST?". We're getting there; what it boils down to is this. The ST has to spend an *enormous* effort to get that screen up there, and is making 17 million dot decisions per second. That's taking up an incredible part of the ST's computing capacity – and is one of the things that directly affects the SST accelerator, as you will see.

...Than To Fade Away...

If you're puzzled by that last title, it's from the song "Rust Never Sleeps", by Neil Young, ("better to burn out than to fade away"), and Neil is my programming music. Remind me to tell you that story sometime!

Well, let's say the ST *does* manage to vary that electron beam in amazing synchronization with where the monitor has put it, and we actually paint a desktop picture, icons and all, one line at a time, with that one tiny little dot. It isn't enough. That phosphor on the screen is *determined* to fade away, too. So this painting process of tracing the screen with the electron beam, over and over and over, goes on continuously by sheer necessity. If it ever stops, your picture snaps out and the monitor goes dark in 1/30th of a second.

Optional Lab Experiment: To see how fast the phosphor goes away, just pull the monitor plug out of your ST while it is running. Zap, the picture is *gone!* Pretty fast, huh?

Hence, you see, the entire screen isn't sprayed out at once by that electron gun at the back of the picture tube. Rather, the tiny electron dot is swept left to right, 400 times, working its way down the screen, with varying intensity, to paint the picture. It takes 1/70th of a second to go from start to finish, and the whole thing has to happen over and over for you to see the screen for more than a brief flash.

With a phosphor fade-time of 1/30th of a second, a refresh every 1/70th of a second keeps the phosphor well lit up. If you push this, you get what's known as "flicker", which is very irritating to the brain.

One reason you don't perceive the image as being made of little dots is that two lit-up dots tend to blur together, and a whole row of them blur nicely into a "line". You'll have to look very carefully at your monitor to distinguish individual dots.

Other Editor: Programmers do this all the time; that's why I have thick glasses!

Now, let's take a break while you refill your coffee (or Pepsi (or Diet Pepsi (or beverage of your choice))) and have an... Interlude.

This page, intentionally left BLANK, is dedicated to George Blank, with all our respect and thanks.

This page is dedicated to the fine man who took a chance on a couple of unknown authors, and published our first article, ever, on the "Secrets of Atari Graphics":

George Blank, of Creative Computing Magazine.

Ask *any* writer how hard it is to get that first one published.

Editor: The hardest part was using a typewriter to make each revision. We typed out the entire first article 4 times (each about 20 pages long)!

This was over 10 years ago; the June article appeared authored by "David and Sandy Small" *before* we were married. Fortunately, we remedied that by the July issue. The results? Well, besides Eric, Jenny, Jamie, Fang and PyeWackette? Spectre 128 and Spectre GCR, MegaTalk, and the SST!

George Blank then turned over the "Outpost: Atari" monthly column to us, and we worked very hard disseminating 8-bit Atari information. As usual, David wrote, Sandy edited, and the pictures were drawn using the PLATO network. Creative Computing eventually folded; sic transit gloria munde. It is fondly remembered by us.

George, when last we talked, had returned to his calling and become a minister.

Historical Trivia: Our last "Outpost: Atari" column published in Creative Computing recommended that *someone* do a Macintosh Emulator for the Atari ST!

Editor: I wonder who that could be!

Revenge: The Hard Disk Saga

I'd like to first say that most people regard me as pretty easygoing, and not too serious about life. I'd hate to give you the wrong impression with this story.

But, of course, anyone can be pushed too far. I was. This is my story.

I had been involved in a long dual project developing Spectre 128 and Spectre GCR. Now Spectre GCR is not a trivial program; it's right around 1,500,000 characters (1.5 million keypresses) of code and comments, in many separate files.

Problem was, I was having a difficult time keeping track of what was what. What was the latest version of so-and-so file? Did I even use these files over here anymore? And so forth.

So, over a month's time, I took a hard disk, cleaned it off, and created what I'll call The Commented History Of Spectre.

I traced every file I had from the humble origins of Spectre, back before it was even released and was in Alpha, then Beta Test, then through the 1.51, 1.9F, 2.3K, 2.65, 2.65C, and 3.0 release versions. I traced the incidental files, the Transverters, the keyboard files, the digital sound files. I traced the READ.ME files that went with each revision. I built up folders that contained the complete original source code and release code for each revision, and doublechecked that they made the right output file, when tested. (They all worked!)

I pruned out the duplicates, the files that were no longer used but I hadn't gotten around to removing, the correspondence got put in proper folders... in short, I played Better Homes and Gardens, and created a spotless hard disk. It was a long, dull job.

This was a pampered hard disk. It had been around a little while, so I knew it wasn't in any danger of "infant mortality" (the first two weeks of anything computerish's life are the most dangerous: that's when components and stuff generally fail). I thought perhaps it would be prudent to make a backup. I went into the next room to get a Syquest 44-meg cartridge to backup on to. When I came back in, I heard a slight screeee'ing noise from the hard drive.

With my heart in my throat, I tried to access the very familiar Commented History of Spectre... and it failed. More than once.

In the blink of an eye, the **Show Survival Kit**, containing every useful hard disk tool known, was out. I tried **everything** on that drive, from SUPEDIT to check out low level access (it was dead), to bumping the head by hand slightly to clear stiction (it was spinning up okay), to swapping in new drive electronics from another drive (still dead), to a complete new ST-506 drive setup, just-in-case...

After the day was done, I knew there was no hope. Oh, sure, maybe a professional hard disk recovery outfit could get it back, but I don't want to even think what they charge, and I had no guarantee the data would be all perfect, which was the point of all this nonsense.

It died right before backup... this has happened to me many times. I have become strongly superstitious about this. I've seen too many drives fail at backup time, which destroys not only the drive but an old backup tape (usually) or floppies. I know that something evil lurks within a drive and it senses backup is near.

Revenge: The Hard Disk Saga

I think I've discovered a principle of "entropy" here (the main principle of entropy is that the universe tends towards maximum disorder, and that entropy is extremely powerful). On hard disks, the principle is, the more organized you make things, and clean them up, the more likely they are to crash, and thus restore disorder. Backups would negate the effect of a crash, so they crash too.

I know, I know, superstitious nonsense... except I talk to other programmers who tell me even worse things. One way around this is to only backup a portion of the hard disk at a time. Maybe your 20 meg drive won't think it's worth it to crash over a measly 5 meg partition. Do 4 of them and you're set.

Also, always do multiple backups. Invariably, backup #1 will develop an error. I don't know why, but as long as there's a #2, you have a chance.

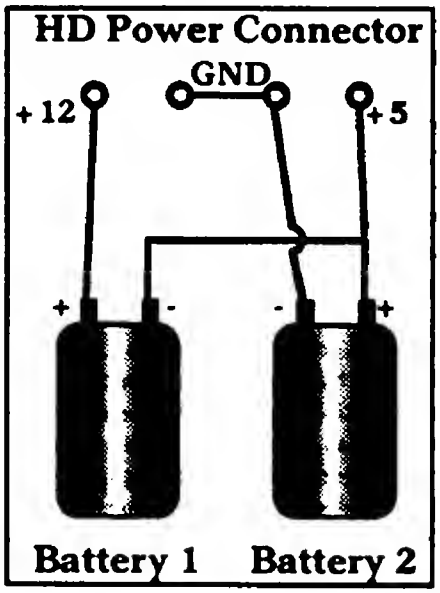
You know, I never did get back to making a clean Spectre archive, and sometimes I wish I had, as I hunt through the hundreds of files, trying to figure out what they are.

Revenge

I woke up one morning knowing precisely what I was going to do with the drive that had eaten all of my month's efforts.

I took two 6 volt batteries, many alligator-clip mini-jumper wires, and a standard hard disk power connector. And some other things.

I drove to the place, and carried the drive out a hundred yards or so. I got out the drive power connector. I hooked up 3 wires, as so:



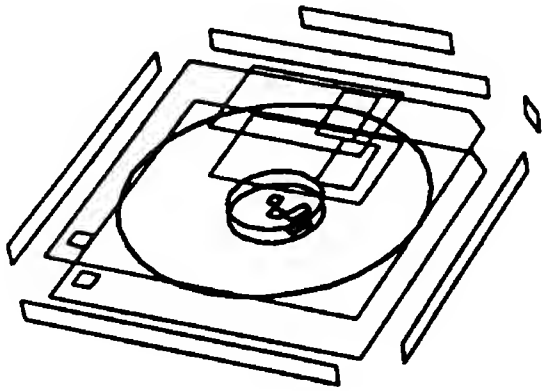
- Drive Ground to (-) terminal of Battery 2
- Drive +5 to 6V(+) terminal of Battery 2
- Minus (-) terminal of Battery 1 to 6v(+) terminal of Battery 2
- Drive +12 to 6v(+) terminal of Battery 1

Hence, what I was doing was supplying the drive's +5 with +6 volts (close enough), and the drive's +12 with twin 6 volt batteries (exact). I'm giving you the exact details for the obvious reason.

I plugged in the drive, heard it spinning up to the usual 3600 RPM, 60 revolutions per second. I set it on the ground with a backstop, with the platter spinning at right angles to the way I walked back. I began whistling a cheerful tune. A dream was about to come true.

Arriving back at where I dumped off the rest of my things, I looked at the hard drive, a ways off but not too distant. I considered a moment, then went with a 7.62 mm size, large, but not so soft the drive's aluminum case would stop it.

I put the rounds into the rifle, and focused the scope on the hard drive. Aimed a little below center; the rifle is sighted in farther than that, it'll shoot high. And with pure, shining anger for what this drive had done to me, I squeezed the trigger. A lot of hate and frustration went into that shot.



The results were spectacular. The drive appeared to leap up into the air about 10 feet on its own, furiously starting to spin, and came down to the ground, still rolling. Then it stopped.

Upon examination, my first shot through the drive had stopped the twin, thick steel hard disk platters, which were spinning at 3600 RPM, in about 1/4 inch (5 mm?) The resulting angular momentum (twist) imparted to the drive frame was what caused the spectacular jumping around. I'd hoped somewhat for the platters to

come apart and go flying (and had selected my firing position appropriately sheltered), but no go. Still, it was good enough.

Then, propping the drive up where it had fallen, I went back, picked up the 5.56 mm M-16A1 rifle, flipped the selector to FULL AUTO, and in one sustained glorious moment, I riddled the damned thing with 30 rounds! It was a wonderful catharsis.

If you're interested in making a major motion picture out of it, I prefer Shwarzenneger over Stallone, really.

I felt, for the first time since the hard disk had crashed, a sense of peace; calm.

I picked up the pieces afterwards. The main frame piece of the drive was still partly together, and I "parked" it over my other hard disks on a shelf. And I said to them, "See this? This used to be Fred. If you ever dare to crash on me, you'll get what Fred got."

My hard disks have changed their ways. Now, when a crash is getting imminent, they get outspoken about warning me. I copy the files off of them and let them die; Abort/Retry always seems to give me a successful retry. I've had ONE out of many Syquest disks go bad (possibly it was the mechanism), and it was careful to check I had everything on it backed up beforehand. It really, really didn't want to end up like Fred.

It really happened this way.

The Blank "Dedications & Thanks" Page (lest this page be left blank ... Horrors!)

There are many people who have contributed to the SST project and to Gadgets by Small who we felt "deserved a mention". There are others that I will doubtlessly forget here, as it is a couple of hours after Midnight as I write this; you know who you are! In alphabetical order:

David & Betsy Ahl, for Creative Computing, who gave us a chance in 1981 with "Outpost: Atari", and who helped spark off the microcomputer revolution. You deserve more credit.

Jim Baen (Baen Books): we got one published, still working on White Sorcerer! A fine man.

Sandy Heidlebaugh, who became Sandy Small, and made my life the many wonderful things it is, and Eric, Jenny, and Jamie, our family, already pushing the envelope. I'm proud of you.

Jimmy Hotz, brilliant hacker, musician, and friend, and his wonderful family. Long will his Hotz Box grace our living room.

Tom and Liz Hudson, just married, and creator of many tools used in the SST's development... I liked your name in the "Terminator II" credits, Tom. NICE! Say "Hi!" to Gary, Jon, and Jack for me, ok?

Gary Hudson, who is trying his best to get us all a reasonable way to orbit: may your SkyRocket Project work out even better than Chris Columbus' did! Luck!!

Elton John & Bernie Taupin for "Rocket Man"; Douglas Trumbull for "Silent Running", for showing me in 8th grade there really is magic, I can only wonder what I'd be without you.

Dan Moore, a true friend, who contributed much to Spectre (like... the whole Menus of the startup page?), and who wrote many articles with me. Thanks, pal.

To be continued...

The SHIFTER Strikes Back

Introducing Video Memory

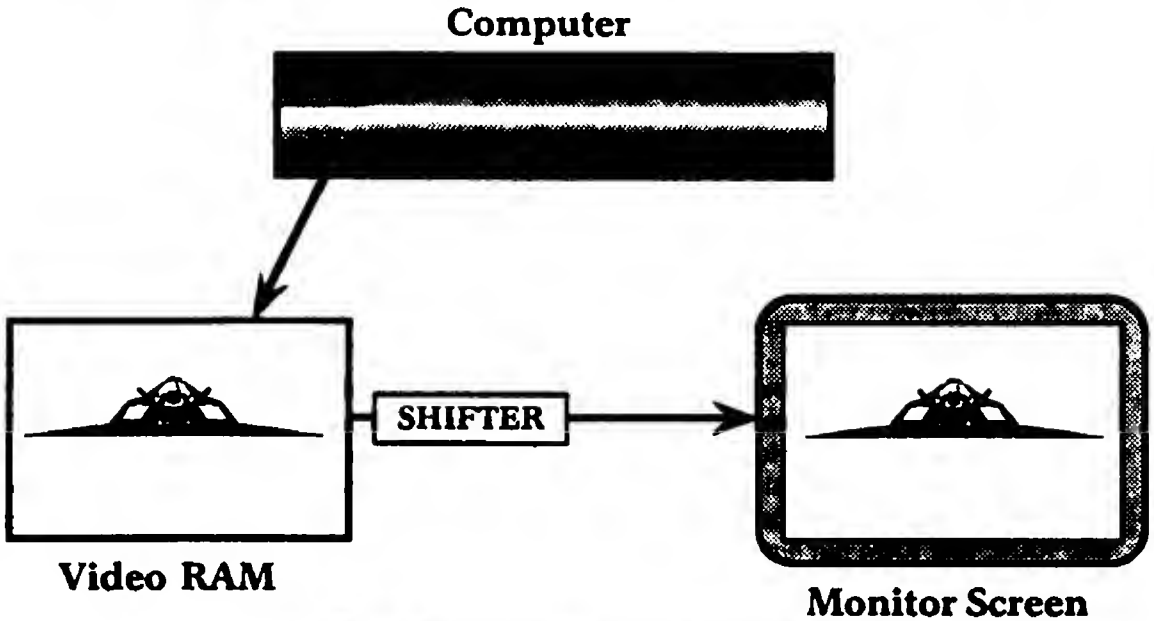
Video images are just one dot being flung across and down the video screen at incredible speeds, based off information fed from a computer. How does the computer do this?

Well, basically there's a wire from the computer to the monitor, where the computer varies a voltage from ON to OFF (1 or 0); that wire controls the electron gun. In color, the computer handles the intensity of the three base R-G-B (Red-Green-Blue) colors; they have more values than just ON **or OFF.**

Now here's where it gets heavy: **How Does a Computer Do This?**

No, you do *not* need a degree in Electrical Engineering or Computer Science to understand this.

See, there's a *lot* of information on that screen, and the computer has to *remember* it in order to display it over and over; your ST monitor can't remember dots by itself, so the ST remembers in its memory. Computer memory isn't very hard to understand; we'll get to it in a bit. In the meantime, think of it as something you can store into and read from, like a floppy disk. Anyway, we come to a section of computer memory called "video memory",

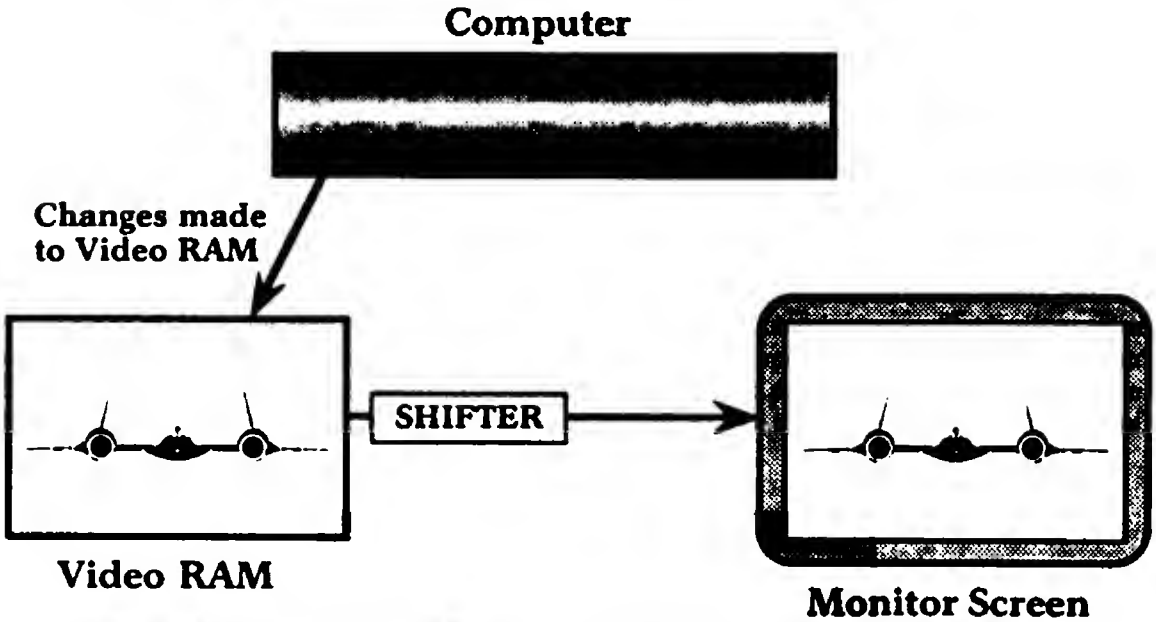


or "video RAM" (RAM is an acronym for readable, writable, good-old-memory. What the words stand for is confusing). Video memory is where your computer stores the displayed image, and where it reads the image to display every 1/70th of a second. It takes 32,000 pieces of RAM to hold the ST's video screen.

You have a lot of things in your ST's memory - the operating system, your program, the data you have typed in, and video memory. In video memory is kept a detailed picture of what should go on the computer screen. When the monitor needs to output the display again, the computer goes to video RAM, grabs the picture (32,000 pieces of memory), and a special chip called "SHIFTER" feeds it to the monitor over the next 1/70th of a second to draw the display. And it does this over and over, forever, while you have a video picture up there. (Illumination 5, previous page.) We'll talk in more detail about SHIFTER in a minute.

If you've programmed the ST, you may know of the PEEK (look at) and POKE (change) commands; these let you directly alter memory, any sort of memory. Should you change part of video memory by directly POKEing in a new value, your video picture *will* change!

Now I need for you to make a conceptual leap. When the Atari wants to draw to video, oh, let's say, when it wants to display a disk directory after you double-click on a disk icon, *it just writes the changes to video memory!* It *does not* draw to the monitor directly. (Illumination 6.)



Illumination 6 - Changing the Picture on the Monitor, by changing Video RAM

When the Atari changes video memory, it effectively changes the display you *see*, though; because that display is being redrawn *from video memory*, 70 times per second. To our (slow) eyes, it's an instantaneous change.

I emphasize this point because it is one people have a lot of difficulty with. The plain truth is that to the Atari, memory is memory is memory. It doesn't care *what* memory is used for by the outside world. (In fact, judging by my debugging sessions, it doesn't care about the outside world at all; it's really an introverted computer.) All the Atari is doing to change the picture you see is following its directions ("program") to write some information into RAM; that RAM *just happens* to be video RAM. When it writes data there, the next time the monitor needs to be refreshed (always pending, 1/70th second later, and almost always in-progress), that data shows up as video dots.

To a computer, *everything* is numbers. Even letters ("ABC...") are represented as *numbers* (in this case, 65,66, and 67 - there is an industry-wide agreement, called ASCII, about what numbers represent what characters). Those beautiful graphics of Dungeon Master™? They are simply numbers inside the "Video RAM" of the ST. That fantastic novel you're writing on your ST? Every character is represented by a number; when the computer wants to print an "A" in your novel's printout, it literally sends a "65" to the printer, which then spits out letter #65 - an "A". And your novel is saved as numbers, one per character.

Yes, folks, pictures, sound, characters, word processing - *everything* inside the computer *is just a number*. All the computer does is shuffle numbers around; that's all it *knows* how to do. The point is that if you make numbers represent other things, the computer can do non-numerical work, while still thinking it is working with numbers.

This is another very difficult concept to get across to people who are starting out with computers. Generally, when I'm showing this at a demonstration, I take numbers that people shout out, and I write them into Video RAM. And magically, 1/70th second later (instantaneous to our eyes), dots appear on the screen. Those are the numbers as Video. I take the numbers and show what they look like as characters; I take the numbers and show what they look like as music (via MIDI or the ST's internal sound chip).

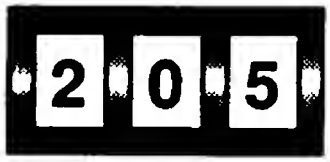
The "numbers" part is the key, though.

Now memory in the ST is stored in units called, incredibly, "Bytes". I don't know who came up with this term, but they deserve flogging, because it's confused people ever since. When you ask, "What's a Byte?", Computer Professionals give you lectures on Base-2 theory... foo! Woof! (For definition of Woof!, see page 3.) If you want to know a little about the twisted minds of computer professionals, get this: they've named half a byte a "Nybble". (No kidding; see the Apple][™ technical manual!)

Dave's Easy Byte Definition

Ruthlessly skipping the whole Base-2 number theory, try this definition of a byte on for size: A byte is one storage space (of many) in a computer, capable of holding up to a certain size number, that can be *individually accessed* - like the address of a mailbox on a street.

Imagine you have three spaces in which one digit, 0-9, will fit (kind of like the odometer on the right). You can write a number from 000 to 999 in there, right? But no bigger; *there is no room for a fourth digit*. A computer byte is *very* close to this, except that because of some computer voodoo, you can only save a number from 0 to 255, inclusive.



"3 Digits"

Other Editor: I know, I know, 255 is a weird number; I can't explain it without going way past where you need to go - straight into Base-2 theory. In fact. Why do you think Computer Professionals act the way they do?

The ST typically has millions of bytes of computer memory in it. The basic ST, the 520 ST, has 520,000 or so bytes. The biggest ST has 4 million bytes. The SST has 8 million bytes, which add to your ST's 4 million or so for 12 million total. Each of these bytes has their own place; a numbered "address" (just like you have *your* own address). This concept is important for later.

Believe it or not, this tooth-fetish byte stuff gets worse. What does a Computer Professional call a million bytes? Why, a "MegaByte"! That's why people call a 4 million byte Atari a "Mega 4", or "four-meg" computer. A thousand bytes is "1K", pronounced "One-Kay" (one Kilobyte).

Other Editor: And how many Computer Programmers does it take to change a lightbulb?

None: that's a hardware problem!

As I said earlier, computer numbers can represent video images, so another way to look at a byte is that it is 8 video dots stored in computer memory, each dot lit up or dark. (More tooth-fetish computerese: each dot is called a "Bit" for Binary Digit.) If you work it out on paper, there are 256 *possible combinations* of ON's and OFF's, given 8 bits to work with. That is why we can save 0 - 255 in a byte; 0 itself counts as one possibility. (We need zero far too badly in computers, which is why we don't save 1 through 256.)

Back to that idea of a byte holding 8 lit or dark dots (ON or OFF bits)... hmmm, that's sorta handy for video, isn't it? We were just wondering about where and how the computer stores our video image... let's store it a byte at a time. (As you can tell, I am forcibly restraining myself from making puns!)

Let's see. An ST has 640 dots across. At 8 bits per byte, we thus have 80 bytes per scan line. 400 scan lines at 80 bytes per line is precisely 32,000 bytes of memory, or 32K.

And that indeed is how it is done. It takes 32,000 bytes of memory (32K) to hold a video image on the Atari, either monochrome or color. (You may know that color has fewer pixels/dots on the screen; the reason it still takes up 32K is that the Atari has to remember RGB color information, not just a simple ON/OFF, which takes up more bits per dot). Atari uses a very, very good scheme to manage video memory, for the truth of the matter is that *driving a display beats a computer to death!* Many manufacturers can only come up with kludges (messy video that blurs or flickers), because they can't handle the terrific strain on memory that video makes! Others do things like dedicate expensive RAM chips *solely* to video, which you pay out the nose for.

Atari has achieved the very near impossible in making an *inexpensive* computer do great graphics. Sure, it's easy to do graphics when you can afford memory chips that are superfast and dedicated to video, and expensive video driver chips - all of which cost the consumer around 4-8 times what they cost the manufacturer. The price of a 1040 ST ($399 or so at this writing), with a million bytes of memory, is very impressive, given its rock-steady, flicker-free, really pleasing 640 x 400 monochrome and 640 x 200 or 320 x 200 color displays.

The Incredible Burden Of Video Memory

Let's do a little math (just a little - keep reading, it's just to illustrate a point). We know it takes 32K for Atari to keep the screen image stored. That's output 70 times a second. So 32,000 bytes output 70 times a second is 2,240,000 bytes per second - call it 2.24 megabytes (million bytes) per second.

That's fast! That number will awe *any* "Computer Professional".

I'm sure you're a little Future Shocked at computers in general and how fast they are. But that 2.24 megs/second is a darned impressive number! Let me briefly give you some comparisons. First, that means filling up an entire Mega 4's memory in under 2 seconds. *Wow!*

There is now (Late 1991) a Mac II SCSI hard disk controller whose ad proudly states, "Can transfer 3 megabytes per second!". It *costs* and requires specific hard disk drives that can handle that sort of (considered awesome) speed. Heck, Atari's near that speed just in video! As for me, I published a little program called Twister that made floppies run at their absolute maximum safe rate. (It's now an industry standard.) This gave me 1 megabyte *per minute*, not 2.24 Megabytes *per second!*

Editor: He said modestly.

And Atari had theirs in *1985*... But you've got *other* problems, too, relating to timing.

Remember that the monitor is mercilessly marching across scan lines, expecting the computer to keep up, and if the computer does not, oh well; the monitor will say, "Not my problem, bud!". The monitor is certainly not going to care if the picture smears or goes slaunchwise because the computer was too slow. The computer has no choice; it *must* keep up with the monitor!

This is why the *number one* priority of the ST computer is hammering out 2.24 million bytes of computer memory to the monitor *every second*, day in, day out, with exact timing. This video-output must go on all the time; the ST doesn't flash-output the video screen in an instant; rather, it outputs it steadily, one scan line at a time, taking 1/70 second for the whole screen, then does it over and over again. The ST is *locked* to the monitor speed, because that's the absolute priority.

I hope you're catching between the lines my personal admiration for the people who managed to design this computer. For all that's been written about the ST computer, very few writers have understood just what an unbelievable task it has to manage.

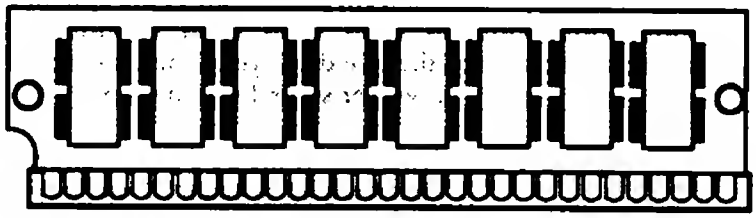
Let's learn a little bit more about memory, particularly since the *speed* of computer memory is the *limiting factor* for accelerators.

Static and Dynamic RAM

Computer memory today is of two types: "Static RAM" and "Dynamic RAM". (RAM stands for Random Access Memory, meaning you don't have to start reading at 0 to find out what information is stored is at 100.) Static RAM is fast, *and also expensive*, to the point where if it is used at all in a machine, it is used in very small amounts. Even the Mac IIfx only includes 32K of Static RAM in it - and that's a machine that sells for so much that Apple could afford almost anything parts-wise for it. I know... my wife likes the speed of her IIfx for her work. Oh, do I know. (My wallet shrieks when I get close to her workstation).

Editor: He's just jealous: I've never heard his wallet shriek!

Static RAM is what you *think* computer memory RAM is. You write something into it, it stays there indefinitely (at least, until you turn the machine off), and you can read it anytime. It's always available and always fast.



Illumination 7 - One Megabyte SIMM

Then we have the sad, dreary reality: Dynamic RAM (DRAM). This is the RAM you're *really* using. It forms the main memory of your ST, and is nowadays commonly packaged as "SIMMs" (Single Inline Memory Modules; Illumination 7) which the ST^E, Mega ST^E, and TT uses. Older ST's (like the Mega) still use DRAM, it just isn't packaged as a SIMM. One megabyte SIMMs (1,000,000 bytes) cost about $40 or so. (Note: chip prices are as volatile as oil prices; I am writing this in late 1991.) DRAMs have come a long way in a few years; I can recall when a 16,000 byte DRAM board for the 8-bit Atari cost $199.

Dynamic RAM is a devil's bargain. It is a bunch of, frankly, leaky tanks that hold bits of electricity (for you techs: capacitors). You write into them, and they're filled with electricity, whose only goal in life is to leak away *really fast*. They leak away unless you pour in more electricity many, many times per second. This pouring in is called "Dynamic RAM refresh". If I recall correctly, about 18 times per second is thought good. In order to refresh DRAM, you have to *access* (read or write) it, which sort of forces the RAM chip to replenish the tank that is accessed, in a weird sort of way. I wasn't kidding about "devil's bargain".

(As you can see, I am translating a great many technical concepts into English here...)

What's *good* about DRAM is we can put a whole lot of it into a itty-bitty chip, since it's just tiny capacitors, as opposed to the big, complex circuits ("flip-flops") that Static RAMs have to use. It's also why you still don't see "1 million byte" Static RAMs; they cost too much and take up too much space.

This DRAM "leak away" means you have to access *a lot of memory very often* to keep it from losing data. This is no joke; "refresh" can positively kill performance of a machine with a bad design, because it takes memory access "offline", away from the computer! It's cast in concrete: if your ST (or virtually any other modern computer) stops refreshing its DRAM, it gets amnesia, and its memory dribbles away bit by bit. Even sitting "idle", at a desktop, the ST is struggling not to lose its marbles!

The *only* good side to all this hassle with DRAM chips is that they are *cheap!* If you can manage to tolerate all their idiosyncracies with your design, your computer will cost far, far less than with a static RAM computer. DRAM is now incredibly cheap, and will continue to be used in the foreseeable future.

A Stroke Of Genius: Shifter, MMU, DMA, and DRAM Refresh

The story goes, in late 1984, the new owners of Atari decided to design an absolutely lowest-priced 68000 based computer. They called it the "RBP", for Rock Bottom Price. They wanted to start with 520,000 bytes of memory, but be able to work up to 4 million bytes. Fable? In fact, I have a spec sheet labelled "RBP"!...

When the designers sat down (with their Jolt Cola in hand) to design the ST, they had all these problems in mind. They had to have lots of RAM. They had to refresh it. They had to shove out an incredible 2.24 megabytes per second to do decent video. They wanted fast disk access, where the disk could directly load to RAM, and whatnot. And do it at Rock Bottom Price.

The biggest problems, as you've seen, are video and refresh. And here's true genius: they used the problems to solve each other!

What they did was organize video memory so that every time a video screen was fully drawn from DRAM, *all* of the system's memory was refreshed, automatically, every 1/70th of a second. **Thus the video problem *solved* the refresh problem.** You might say that the displayed image is the only reason that your ST is keeping its marbles! (A philosopher's mind would boggle!)

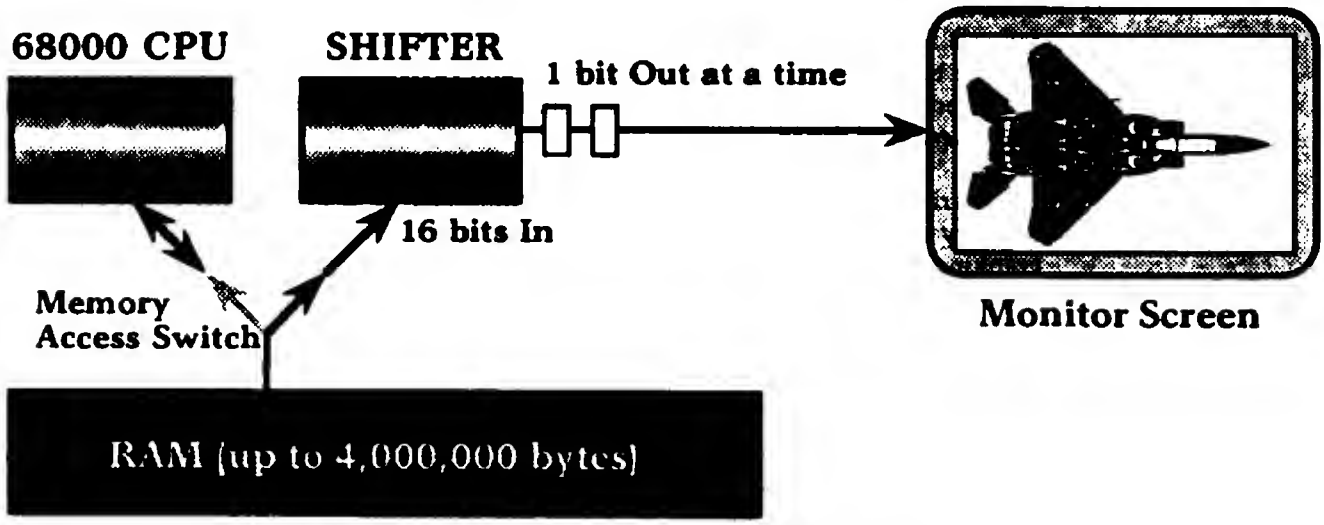
This part of the ST's design directly affects the design of the SST. We *must not foul up* ST video output, or main ST memory will stop refreshing and collapse. This is a *key design decision* in the SST accelerator, as you will see shortly. Now, the computer's CPU (Computer Processing Unit), and video have to share memory. The designers shrugged and decided a 50-50 deal worked out pretty well. So, in the ST, the CPU (which just happens to be a 68000 microprocessor) and video *share* the computer's RAM by taking turns; this sharing is well enforced by other special chips (we'll get to them later). CPU and video are not allowed to trip over one another's feet.

I hate having to do this, but I need to introduce a convenient term for Millions Per Second: it's called megahertz, pronounced "Mega-Hurts". (No comment.) [There was this guy named Hertz that helped invent radio, so it's named for him.] It used to be called megacycles, which us older radio/electrical engineers remember. Megahertz is abbreviated MHz by *everybody*; you will see it listed in just about any computer ad, as it is the basic raw speed of the machine, like MPH or KPH is for cars. And, as we'll see, it's very simplistic.

Other Editor: And if I don't start abbreviating, this manual will end up 200 pages long!

Remember when I said, awhile back, that the ST is hammering out video at 2.24 million bytes per second? That's 2.24 MHz. (See how easy it is to be sucked into using jargon?) I have simplified video somewhat; we actually need more than 2.24 MHz to handle video.

Memory is rated in speed, in the number of times per second you can *do something* with it. What Atari did was put 16 Megahertz RAM (which can do 16 million "things" per second) into the ST, and split the RAM's functionality, dead even, between the CPU and video. The CPU, a 68000 microprocessor chip, and SHIFTER, the video chip, therefore *each run at 8 MHz*. This is a tremendous amount of access power to give to video, but as we've seen, the ST needs it to keep that image up there on the monitor. (Illumination 8.)

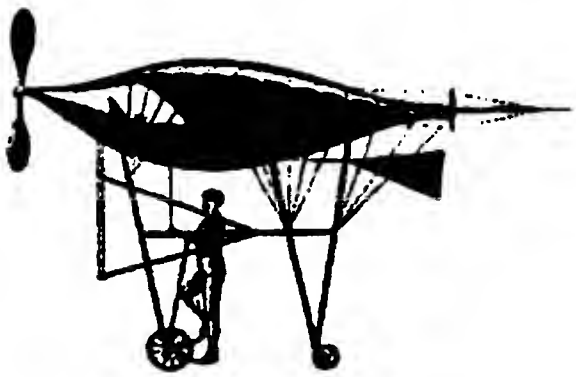


Illumination 8 - Sharing RAM between CPU and Video

Basically, the memory alternates between the 68000 CPU and the video, giving the 68000 odd numbered cycles and video even numbered cycles. Memory goes, "CPU cycle, VIDEO cycle, CPU cycle, VIDEO cycle"... and each chip can *only get access* during its cycle. (If they try in the wrong cycle, they have to wait their turn.) Sort of like Dr. Jekyll and Mr. Hyde!

A 68000 tends to access memory every other cycle when run at 8 MHz. So does SHIFTER! Thus, the CPU in the ST can run full speed at 8 MHz, and the video can run full speed at 8 MHz, and the 68000 and SHIFTER *don't fight with each other*. In fact, they're not that aware of each other. Quibble: They conflict a little *tiny* bit, but the 68000's internal timing works out *incredibly well* for this setup; when you run a 68000 chip at 8 MHz, it asks for memory every other cycle, which fits in nicely with the "CPU cycle" of memory.

This is another factor which directly affects the SST design. *THERE IS NO WAY TO GET TO ST MEMORY FASTER THAN 8 MHZ* - that "CPU cycle / VIDEO cycle" circuit design enforces it. As you can imagine, to an SST with a 68030 microprocessor, even running at 16 MHz, Atari memory is S L O W...



This picture is Dave's Mom's idea of a good visual image of Atari Slow RAM. I liked it!

On the video cycle, the video chip, with the aid of various other chips like the MCU (Memory Control Unit), grabs some memory and stuffs it into the "SHIFTER". The SHIFTER is one of the big, custom chips on the ST. It's called SHIFTER because it

takes this video data, in the form of two bytes: 16 dots (or bits or pixels; whatever you want to call them), and one by one, "shifts" them out into the monitor at the exact right speed. (The shifting is something like a row of 16 seated people standing up and moving one chair to the right, with the firstmost person going out the door - sort of like a doctor's office line. See Illumination 8, previous page.) The MCU is sometimes mistakenly called MMU, which is *not* the same as a 68030 MMU, by the way.

On the 68000's cycle, the memory is used for many and varied functions - storing and retrieving programs, data, video, sound, and whatnot.

Neat, huh? The designers made the disadvantages of the ST's design *help each other*. For instance, even though video is an immense strain on the ST, it helps refresh memory! Back in 1984 when the ST was designed, the Common Industry (cheapest) 68000's usually ran at 8 MHz, so the Atari ST designers took the times when the 68000 was not fast enough to use memory (every other cycle), and gave them to video. Video got the priority it needed to keep up with the monitor, and the CPU wasn't interfered with. As I said, it was a *very good design*.

Atari then put the design into custom chips such as SHIFTER, MCU, GLUE... (which are very expensive to design and lay out, but once done, cheap to make), and started cranking out ST's. At some point, the cost of making the chips was overcome and Atari started making money.

So, you see, the big fight in the ST isn't over the CPU speed at all. All the fuss is over *memory!*

Now, I don't know about you, but my coffee's gotten cold. Time to go microwave up another cup of water (80 seconds on my microwave), two scoops of Folger's Instant, stir, and return to the word processor. Which can only mean...

Editor: Pepsi break!

Big Time Video Fun

Seems like everybody's got a video camera these days.

Whenever I give a talk at a computer show, there's inevitably a camera taking it all in. Talk about adding to the pressure... and worse, I can't imagine why anyone would want to tape me. At birthday parties there are multiple cameras. At Eric and Jenny's last day of school, there must have been 50 cameras. I was amazed. People were even video-taping other people video-taping!

Nearly all video cameras have the ability to plug straight into a TV, to give you "live" video coming out of the TV. This can be fun; if nothing else, it shows you how very difficult it is to make people look good on video, and why TV News Anchorpeople get paid so much. (It ain't brains, it's makeup!)

Now let's do something interesting. Hook up your camera to your TV, hit RECORD so it's sending a signal to the TV – **and aim it at the TV.**

A most amazing image will happen. If you've ever been between two mirrors, and seen the "endless hallway of mirrors" effect, you'll know what I'm talking about.

What's happening is the TV is feeding the camera, which feeds the TV its own image, which feeds the camera, forever.

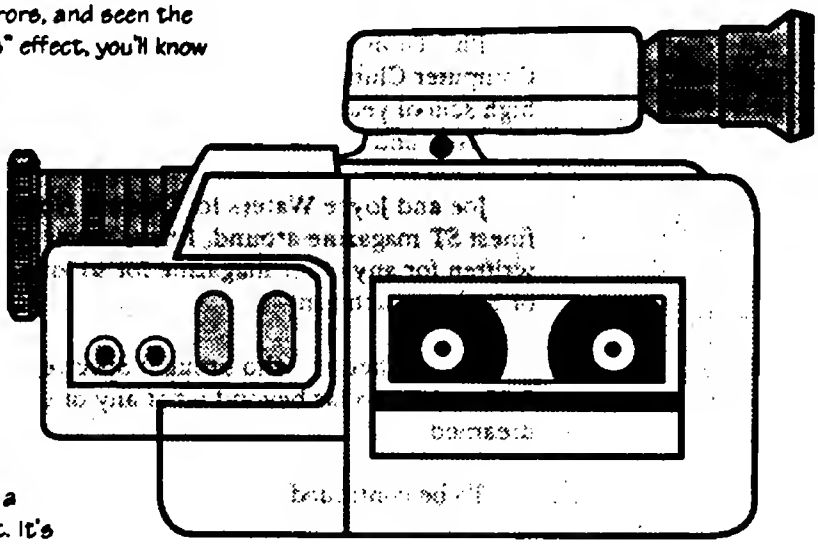
Now try rotating the camera on its side slightly. The image on the screen will slowly "ripple" into the center, rotating more each ripple; you'll get a **fascinating** twisting effect. It's really wierd! Also try zooming in and out for even more wild effects!

If you have kids, put them next to the TV, and suddenly there will be a hundred of them in the endless video hallway. Kids get an enormous kick out of this; in fact, one day Sandy and I were desperate for something to entertain the kids (while we had a teleconference), so I hooked up the video camera, showed Eric this trick, and off we went, down to the basement. Two hours later they were **still** at it; kids are natural camera people, they haven't learned to be camera-shy yet. (We're saving the tape for when he brings home his fiance.)

Technical people call this "Video Feedback", and it is now one of many "Special Effects" that are used in TV production. But you can try it out with your very own video camera.

I call this "Video Baby-sitting". Check it out!

Editor: Lights! Camera! Action!



More Blank "Dedications & Thanks" Page (lest this page be left blank ... Horrors!)

Jerry and Roberta Pournelle for their consistent, warm, long term support of our family business, and Alex Pournelle, longtime friend.

Kevin Reynolds & John Milius for some really good movies.

George Richardson, who designed & debugged the SST and MegaTalk. (What more can be said?) A fine man with a fine family.

John Ridges, for his fine knowledge of disks and the GCR. I believe John is overall the finest combination of software and hardware engineer I have ever known.

Gene Roddenberry, another "dreamer" ... like me. A man who changed the world.

Tom Scholz for defining the guitar sound I so love with his Rockman™ (see Introduction) and for the best rock album side ever done: Boston, Side 1.

Sheridan, Ross, and McIntosh, George, Todd, and Tom, thanks. You deserved the Dom Perignon.

Jim & Carol; thanks for hanging in there.

Nikola Tesla, the finest hacker that ever was. May more people learn how much they owe to you.

Phil Tubb, for the Jefferson County Computer Club, who did so much for my high school years, along with John Ridges, Tim Gill, and others.

Joe and Joyce Waters for Current Notes, finest ST magazine around. I have never written for any other magazine for so long or had so much fun!

Doug Wheeler, who brought Spectre 2.65 and 3.0 to far beyond what any of us dreamed.

To be continued...

The Accelerator Rebellion triumphed, and they lived happily ever after, real fast...

The Revenge of the 68030

I realize it took some time to get here, but, essentially, until I showed you *why* the ST is so tightly locked to video speed, and why memory is locked to video speed, and why CPU memory speed must be 8 MHz, there was *no way* to show you how an ST accelerator *must* be designed if you want pure speed. These details are really important to keep in mind while I explain the SST's design.

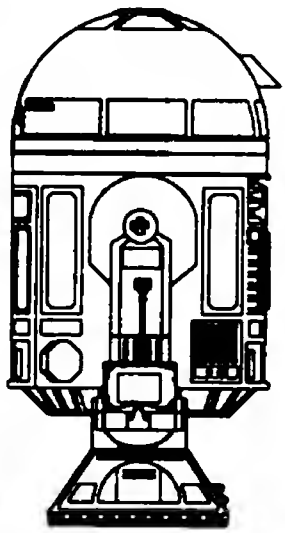
The Plot so far: The ST has 16 MHz memory, split 50-50 between the CPU and the Video. The CPU is strictly limited to 55 MPH... err, I mean 8 MHz access to ST memory. Video repaints the screen 70 times per second to keep a display going and *must not be stopped*, otherwise all of ST memory will collapse for lack of DRAM refresh.

Hey, check it out... look at all the buzzwords you know!

Editor: Gee, Dave, you make it sound like a movie or something: "Attack of the Beings From Planet X". Which, as everyone knows, is the last known source of Illudium Phosphate, the shaving cream atom. Must be why Dave is growing a beard!

Enter the Accelerators

Your computer executes a set of instructions, called a program, like this. A 68000 instruction is two bytes, sometimes followed by some more bytes if it's particularly complex. The 68000 grabs the 2 bytes, does what they say, grabs the next two, does what they say, and so forth until you shut it off, going about 700,000 instructions per second, on average.



The "two bytes" (16 bits) works out nicely, since ST memory is organized to give you two bytes anytime you ask for anything (even if you ask for one, you sort of get two; the other one is ignored.) Remember, video *also* works two bytes at a time.

One day, someone got a bright idea, and stuck a 16 MHz rated 68000 into the Atari ST, replacing the 8 MHz rated 68000 that is presently there. They then changed the "clock", the "heartbeat" (the oscillator which sets the 68000's speed), from 8 MHz to 16 MHz, and confidently expected the speed of the Atari to increase to twice what it was.

Whups. The experiment flopped. There was maybe a 10% increase in speed - and they were expecting a near doubling, a 100% improvement in speed.

What happened? You can guess from the above: they ran into what computer engineers call "video contention". Video (the SHIFTER) kicked 'em out of memory!

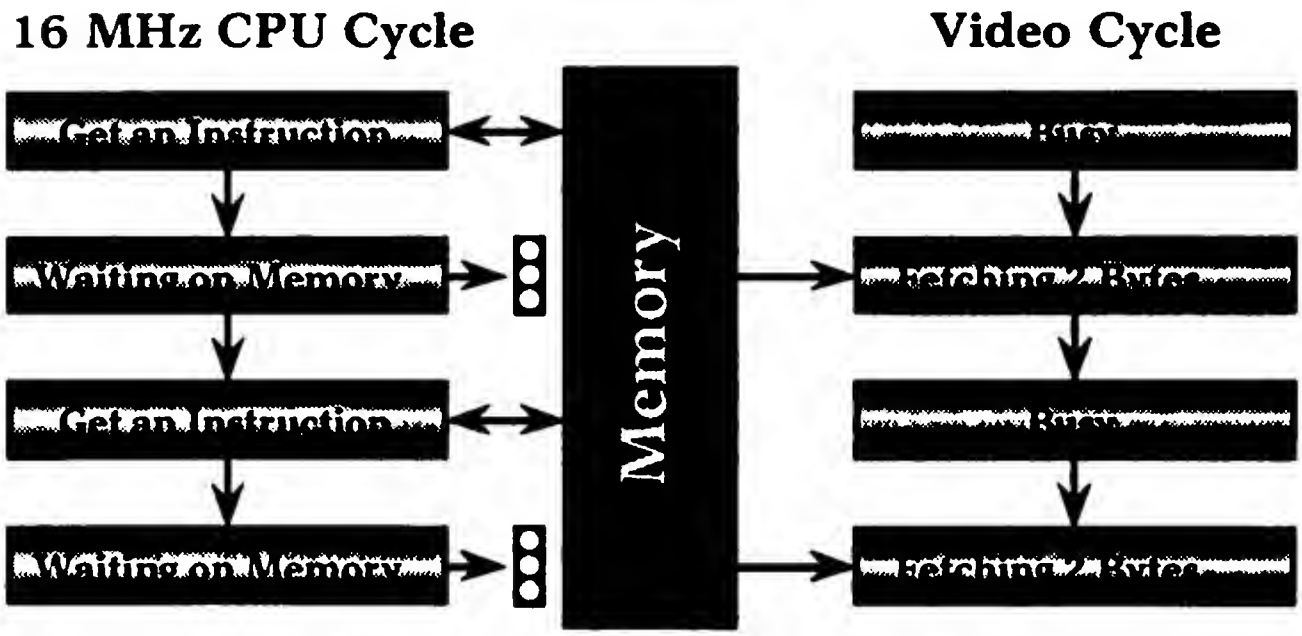
You see, a 68000 works by "fetching" instructions from memory, and executing them. Now, at 8 MHz, the 68000 rarely noticed video; mostly, video and the 68000 stayed out of

each other's way. When they tried to step on each other's toes, the other ST chips enforced the rules. But not so with a 16 MHz 68000! That CPU is running twice as fast. It would like to get instructions from memory twice as fast. Since ST memory is basically 16 MHz, that CPU could hog memory *all by itself.*

Of course, *that* would be a disaster - since if video can't run, ST memory will not be refreshed, and will collapse. Then where will the 68000 get its program?

The ST's video circuit, faced with this threat, calmly shut off ("wait-stated") the 68000 whenever video needed memory. *Video Gets Priority*, always. The 16 MHz 68000 was limited to 8 MHz instruction feeding (very much like starving an engine for gas). So the "advantage" of going to 16 MHz was almost totally negated, because the 68000 couldn't *get* any new instructions to execute - and the 16 MHz 68000 spent a long time "wait-stated", on hold, in effect.

It was the video memory bottleneck. Frankly, if you want something to look at on your screen, you have to give up "8 MHz" of your ST's memory to video. When we looked at the ST, we found there's no *practical* way of shutting down video, or making its load less onerous to the ST. Atari's custom chips cannot be changed, and it's extremely difficult to even intercept wires heading for them to alter functions. The GLUE and MCU chips are out there stuffing video to the SHIFTER, and they don't care that they are getting in our way. And, of course, they get priority - the monitor can't wait (and neither can the DRAM). (Illumination 9.)



Illumination 9 - A Faster CPU can't get to Memory Faster

The only advantage gained, in fact, was in the rare 68000 instructions that do things that don't need anything from memory whatsoever, like shift instructions, which take lots of cycles. Those speed up. And as the 10% overall speedup value shows you, the ST doesn't use those instructions a whole lot "In The Real World".

Cached Accelerators

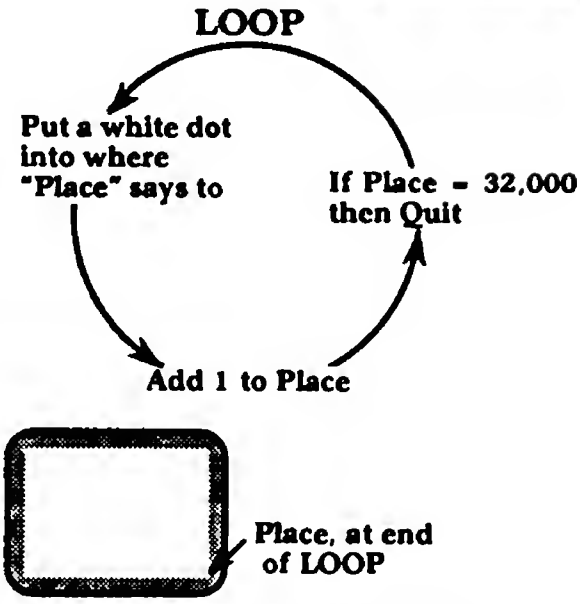
The next big step in acceleration came with "cached accelerators". A cache (pronounced "cash" (no pun intended)) works by lowering the number of times the 68000 processor has to go to ST RAM (and thus be forced to wait by the video guardians). It is a small, usually 16K, Static RAM memory chip; yes, that means it is fairly expensive. (But no refresh is needed, in Static RAM!) What a cache does is *remember* what you read/wrote from/to ST memory last time you accessed it, and if you need that byte again, it comes

from the cache, not ST memory, *without any video slowdown*. It is memory *independent* of ST memory, capable of full blast 16 MHz access with a 16 MHz (*not* 8 MHz) 68000.

Why does a cache help, given that it only handles the *second* and subsequent accesses to ST memory? Because most programs tend to execute the same instructions over and over again; it is common programming practice.

For instance, let's say we're simply trying to clear the ST screen. This is a matter of writing 32,000 bytes of white dots to video RAM. If your program is in ST RAM, and we're trying to run at 16 MHz, it'll have to wait for each new instruction as it does this:

Place = 1st Video RAM place
LOOP: Put white dots into Place
Add 1 to Place (so we're pointing to next video RAM place)
If Place = 32000 we are done; Quit.
Otherwise "Go To" Go Back up to LOOP: and do it again.

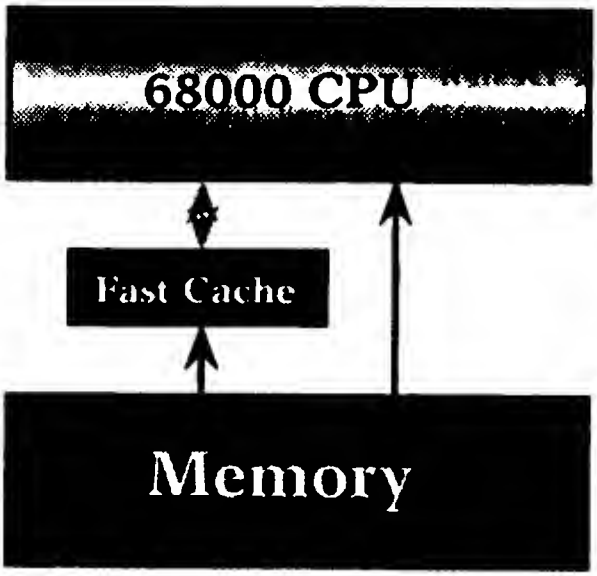


Illumination 10 - A Simple Loop

Hence, there's plenty of waiting time between executing each instruction, even though it's the *same instructions* being executed over and over. (Illumination 10.)

In a cache system, anytime you read (or write) ST RAM, the place you worked with in ST RAM and the value you read/wrote is recorded in cache RAM. (Naturally, since you have up to 4,000,000 bytes of ST RAM and only 16,000 of cache RAM, the cache cannot remember every one! This is a serious problem.) So the cache usually works on a last-access basis; whenever you load a new place and value into the cache, the one that was accessed the longest ago is removed from the cache. The cache sort of figures that the newest stuff is what you're interested in, anyway.

Now, when we first fetch the instructions for this loop, they are slowly fetched from ST RAM, and also loaded into the cache. (Illumination 11.) The cache now "knows" what values are in this program's memory locations! When we go through the loop again, and the 68000 asks for the next instruction, the cache steps in (*very quickly*) and says, "You don't need to go wait on 8 MHz ST memory. I have the bytes you need right here." The 68000 gratefully accepts the bytes, and *doesn't have to wait on ST RAM* to get its next instruction. So, the cache ends up feeding the 68000 at (aha!) 16 MHz! (Illumination 12, next page.) Of course, the write to Video Memory is *always* 8 MHz, because you have to wait for ST RAM, in order to store a value.

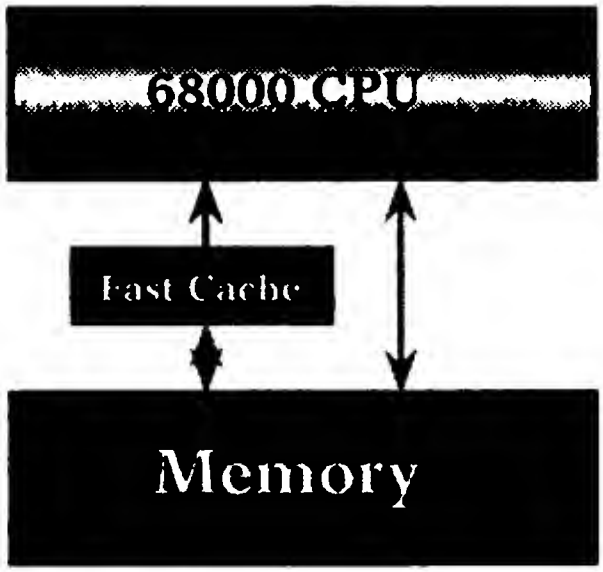


Illumination 11 - If the next instruction is not in Cache, then Load it into Both

Caches can be good or bad; it really *all* depends on the program they are working with. If it has loops, where the cache can "load up" the loop and supply it to the 68000 over and over, that's great! If the program jumps around all over, though, the cache really never gets an opportunity to help; it can only store the last 16,000 memory accesses, and while that may seem like a lot, remember, this is a machine hitting memory millions of

times per second.

I own an IBM clone computer, and it's an 8 MHz machine, just like the ST. I bought a 16 MHz 80386 accelerator for it, with a 16K cache (It would be a lot like putting a 68030 and cache into an ST.). I've found while using it (for instance, to write articles) that some things are sped up, and others are not. For instance, small, fast, tiny loops really zing along. Big programs don't seem to benefit much.



Illumination 12 - Load the next instruction from Cache, not Memory

A "benchmarker", or speed measurement program, says that the IBM clone is 17 times faster than it used to be. That's ridiculous; there's no way it is, overall. However, benchmarks *are* small, fast loops, and a cache works well with those. Remember, for a cache to work effectively, you've got to re-run the same code over and over, and without too much work in between re-runs, or the cache is filled full of other data. Any memory access of the RAM kicks out the oldest thing loaded into a cache.

I have also used a true 16 MHz 80386 computer, no cache, where video contention was not strangling the processor to 8 MHz. It flat *flew* in everything it did, not just small, tight loops. The screen scrolled so fast on a "DIR"ectory command I couldn't read it! (My clone never does that.) It's just a much higher performance machine.

So, how much does a cache *really* help? I wrote a review of 16 MHz 68000 ST accelerators, and tested them for performance in real world programs. The best of them gave a 50% acceleration (in other words, got you to about 12 MHz), the worst gave almost no acceleration. This indicates the cache is working about 50% of the time, in the real world. (That's not bad, for a cache.)

For us, the independent memory idea of the cache seemed the way to go with the SST. The key was "Independent Memory Not Tied To 8 MHz". And what we found, when we unleashed fast, independent memory, was quite a nice surprise!

The SST: Engine Philosophy & Design

Well, one way to do a 68030 board is to put a 68030 and a small, 16K cache on to a small circuit board. That way, you get "okay" performance. Remember our 50% cache-working rate? That's not going to change; so you're going to be alternating between full speed and ST speed.

That left a bad taste in my mouth; I wanted *pure speed* out of an accelerator, or I wasn't going to even bother.

Heck, I *already own* a 16K cached machine. I also own 16K cached ST's, with 16 MHz 68000's. In some things, particularly mindless benchmarks, they claim a full 16 MHz worth of performance. In other things, *like what I do a lot of* (software development and writing), they seem to barely speed up the machine.

I mean, to me, caches were like buying a "ghost accelerator". Sometimes it was there, sometimes it wasn't. And in running programs I considered a fair test (the programs I use), they came out about halfway between 16 MHz and 8 MHz - about 12 MHz performance.

But I thought I paid for 16 MHz performance! This didn't appeal to me at all; I felt it could be done better.

So I began thinking about how to do things right. I spent a lot of time considering how to kick the video circuit out of memory so that it didn't take up all but 8 MHz of ST memory. I never found a way; those ST engineers built that whole CPU-RAM-Video Cycle circuit to work one way, and heaven help you if you get in the way! And I saw the computer industry moving ahead around me – computers with new faster processors, more memory, and such becoming available.

The 68030 CPU: Why the SST Goes Mach 3+

The CPU had to be a 68030, for speed and compatibility reasons if nothing else. The 68000 just wasn't available from its manufaturer, Motorola, rated above 16 MHz, and heck, let's face it, 16 MHz is *not* on the leading edge of today's technology. The 68030 is available in 16, 24, 33, 40, and 50 MHz versions, which sounded just *fine* to me; I wanted enough acceleration to make my ears bleed.

Editor: Dave's always been partial to acceleration, as you can see from his column in Current Notes, 1990:

> *"I mean, imagine sitting in an SR-71A Blackbird, just idling. All this... POWER... is around you. Two big turbofan engines just waiting, idling around you, all under your control.*
>
> *There's just nothing better than sliding the twin throttles forward through the afterburner detents and the exhausts form standing-wave diamonds in the air and you're feeling yourself shoved back into the seat as you accelerate, pull back on the stick, go through twenty thousand feet thirty thousand feet forty thousand feet... what a Rush!... levelling out at eighty thousand, Mach 3."*

The Motorola 68000 series has several chips. The 68000 was the original and great, and much loved, chip. It was and is used in many different computers. The 68010 was a minor improvement for "virtual memory" use and never saw widespread usage. The 68020 was used on *many* machines, particularly the Mac II, Sun-3™, and UNIX™ workstations. The 68030 is more or less a combination of the 68020 and the ultra-powerful 68851 PMMU ("Paged Memory Management Unit" – something I'll get into in a bit) chip. The 68040, the newest and Wow! Is it expensive or what?!?, is a combination of the 68030 and the 68882 Floating Point Unit, plus some added goodies for speed.

The ST market philosophy has always been, Power Without The Price™; we could not see going with the 68040, which cost us megabucks per chip ($1000+!!) at design time, when the 68030 was widely available and capable of amazing performance, at much more acceptable cost.

Using the 68030 turns out to simplify several things for us. It has many powerful features on one chip; that MMU is amazingly versatile and powerful. The original 68000 instructions have been sped up, so everything sings along faster (shifts are 17 times faster!) The 68030 also has a built in "pipeliner" to let you do "burst mode" (awesomely fast memory accesses), and the 68030 "talks" 4 bytes at a time, where the 68000 can only do 2 at a time. Finally, the 68030 had been out long enough that it didn't have any more *known* bugs.

Our Own RAM

At that point, the thought hit me: Why not just *add* memory to the ST? Not a little bit of cache memory (used maybe 50% of the time in real life); but serious, *100% of the time useable* memory, like I saw with the IBM. Make the memory "Not Accessible to Video", so that it's not bottlenecked to 8 MHz; rather, let's run this new memory wide open, as fast as it can go. And while we're at it, let's organize the new memory "wider". In the ST, when you ask for any piece of RAM, you get a "word", or 16 bits (two bytes). I wanted my new

RAM to give me a "long", or 32 bits (four bytes), *each time* I asked for memory. That way, you don't have to do two requests to fetch in 32 bits of memory, only one, which saves time.

Once the decision came to add memory, we thought about what type. We're in a different age, folks, than even two years ago; dynamic RAM is *cheap* and available in great heaping chunks. Static RAM is expensive and available in small, dinky pieces. We chose dynamic RAM, even though it's a hassle in the design phase, *because it meant we could make and sell the SST board for less*. Power without the price. To be blunt, it costs us more design and debug time, but as a result, it costs *you* less money!

There were several reasons for this SST RAM. First, a program running in this RAM would scream with performance. It would have *no video contention*; the CPU could get to it at any time. No 8 MHz stop lights! That is very important; video contention was what strangled the previous generation of accelerators. It would work in 32-bit big chunks at a time, so memory access was twice as fast right off, too.

The 68030 could also take advantage of a supercharging mode called "Burst Mode", where the 68030 pulls in the next 8 instructions much, *much faster* than normal. (You have to design your hardware in a particular way to get it to work; we did.) It is very close to car supercharging; your memory must handle some new things; in burst mode, the 68030 tells memory "Burst Mode Request", and memory must "blow in" 16 bytes, or 8 instructions, without a lot of help from the 68030, as fast as it can, the same way a supercharger blows in a gas and air mixture. However, the performance increase is simply awesome, and well worth it.

At this point, the design was a 68030, with separate high speed RAM dedicated to the CPU only, with no video slowdown (access) allowed whatsoever. ST RAM would be left alone, in order to handle the video as usual, and you would also still be able to run programs from it.

We looked at RAM costs, and were amazed to find that RAM is down to about $40 or $50 per megabyte! Given that we were adding RAM, we looked at how much we could fit in the *physical space* of the Mega ST, and 8 of the "SIMM" (Single Inline Memory Modules) could fit, at 1 megabyte per SIMM. That was 8 megabytes added. Furthermore, it could *co-exist* with ST memory, to give you up to 12 megabytes total; 8 megabytes of very quick RAM (SST RAM), 4 megabytes (or whatever) of the usual 8 MHz ST RAM.

Philosophically, what sold me on this idea was that ST users need more RAM! We're past the days when 4 megabytes was so much that you couldn't use it all. For example, with Codehead's utilities, you can load lots of Desk Accessories; Revolver™ lets you swap between already-loaded programs; sound digitizing programs badly need more memory (digitizing really chews up memory!); page layout programs needed more (especially with scanned/image or color files!); spreadsheet users needed more; RAMDISKs can always use more... as I listened to what people needed, it added up to more speed and more RAM.

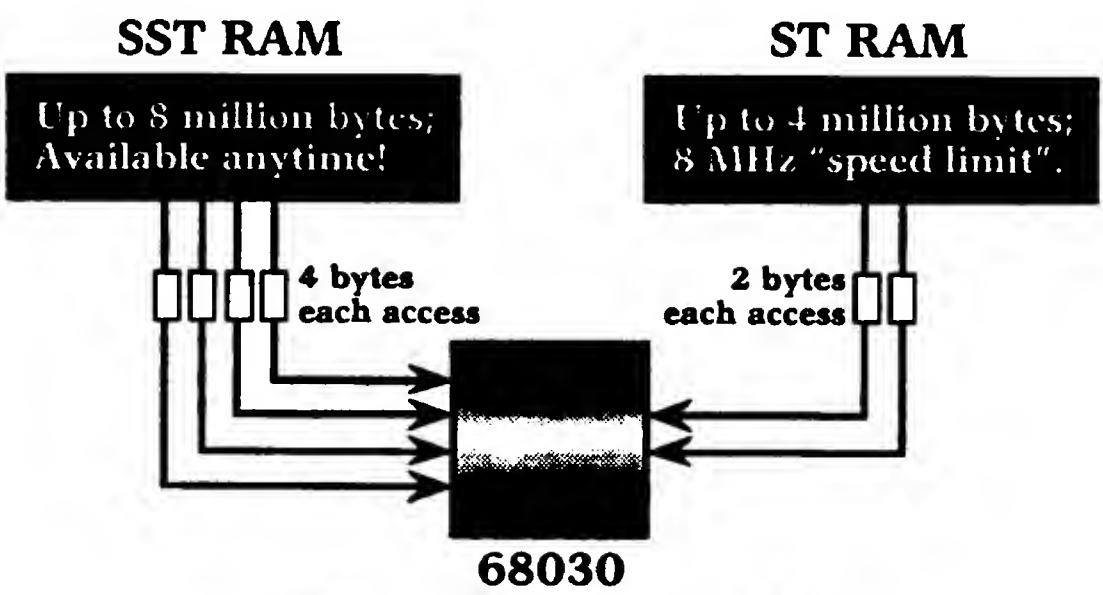
So, we went with 8 megabytes of RAM in standard SIMM sockets. This is *the* type of RAM that is widely used in the computer industry (in both Macs and IBM clones), which means competition has driven down its price (as opposed to some specialized type of RAM). This is the cheapest, most garden-variety RAM you're going to find.

We did good things for you, in enabling this common RAM to run at ultra high speeds that burst mode requires. It used to be standard "industry thinking" that a more expensive RAM, called "nybble mode RAM", was required to do burst mode; we did it with common "page mode RAM", the cheapest you can find. Power without the price.

Other Editor: I also have heard that "industry thinking" says an ST can't run Mac software. Funny thing...

Again, this 8 megabytes of memory *adds* to the 4 megabytes already in your machine. It in no way takes that RAM out of the system. So you sort of end up with a Mega 12 - a 68030 processor, and up to 12 megabytes of RAM.

That's a *lot* of computer power.



Illumination 13 - SST RAM; A Four Lane Super Highway

The RAM is special, as I've noted. We've kept the video access out of it, so it runs at its full speed. ***Nothing*** kicks the 68030 out of this memory. We call it SST RAM, or fastRAM. (Illumination 13.) If you load a program into there, it will run at amazing speed, because there isn't anything slowing it down. As a for-instance, as a test, I loaded 8 megabytes of program into fastRAM and ran it; the program was a whopping 8 megabytes long (4 million instructions). This took under one-half second to execute; the SST was running at a true *7.2 million instructions per second.*

Note that there is a large difference between 68000 instructions, which take awhile to run, and "clock speed" or somesuch. Millions of Instructions Per Second, or MIPS, is a widely known computer standard, and 7.2 MIPS is very hot indeed. Considering that a regular Mega runs at about 0.7 MIPS...

The 4 megabytes of ST RAM is perfectly useable. However, it's *slow*, as we have seen, and there's nothing we can do about that: this is intrinsic to the ST's design. It has video limiting it to 8 MHz. You shouldn't use it unless you have to - let's say, some program that for a weird reason insists on using it. ***You'll get the most speed by far from running a program in SST RAM.***

Now here's an important thing: All video and also disk access must *still* come through 8 MHz ST RAM. We cannot display video out of fastRAM, because we don't want to limit its speed to only 8 MHz. Another design feature of the ST I have not gotten into is called "DMA", or Direct Memory Access; this is how your disk drives read or write into computer memory. DMA can only access ST memory.

However, this is really not a very big deal at all to me. If a program in SST fastRAM needs something from disk, my software reads the data into ST RAM, then flash-transfers it to SST RAM as needed. The transfer loop is, I believe, the fastest possible; it uses 68030 burst mode and MOVEM instructions (heavy stuff indeed). As for video, we always leave it down in ST RAM, since you do want something to look at on your monitor. If a program tries to display video from SST RAM, we print a warning message, so you can tell what's going on.

The performance of the 68030 is so quick that in my view it doesn't bog down disk operations unbearably. Really, I wouldn't be able to stand it if they bogged down; you'll recall I'm something of a disk speed fanatic, from Twister, the floppy disk accelerator, on up. I've had no problem at all with the disk speed of the SST; it's sure not slowing things

down for me.

Math Coprocessor

While the 68000 and 68030 are good at math, they cannot be as fast as a chip designed *only* to do math. There are two such chips from Motorola we included in the SST design; the 68881 and 68882 "FPU" (Floating Point Unit). *Either one* (but not both) can be used.

Software *must* be written specifically to take advantage of an FPU, or it doesn't achieve *any* acceleration whatsoever from an FPU. The FPU uses special opcodes that must be in the program you are running.

In the ST, there was a floating point setup with a 68881 hooked up in a different (slower) way that is not "Motorola Coprocessor" standard. The original DynaCADD™ used that, and will not work on the SST. The TT compatible DynaCADD works with the SST.

In the ST world, honestly, not very much software does this to take advantage of the FPU. There are only a handful of programs, notably DynaCADD-TT™. However, the TT has the FPU, and programs will inevitably be written for the TT that use the FPU, so think to the future.

Should you "make the jump" to the Macintosh™ world with a Mac emulator (like Spectre GCR), you're in for a real shock in terms of speed. I ran a number cruncher program which took 60 hours (!!) on the SST; it was doing billions of calculations *without* the FPU enabled. I enabled the FPU: the program then took just *2 hours*. That's a 30 times improvement.

Well, so far, the SST design was fairly well laid out – a 68030, 8 megabytes of RAM to feed it, FPU if you needed it. What next?

Expansion Connector

One thing common to the *really* successful computers out there is *slots* – places where you can plug in custom boards to make your computer do specialized things. Slots give you a way to customize your machine. The Apple II and IBM machines have succeeded partly because of slots. In the Atari, the Mega ST has one slot, and the TT and Mega ST[E] have a specialized "VME" slot.

We had a tiny bit of room left on the board, so we added a processor-direct slot, the SST Expansion Connector. This slot gives you *full* access to the 68030 and everything else... as far as I know, you could add a 68040, if you wanted to, via the expansion connector!

One purpose the slot is already being put to is a video board! Imagine a video board not dependent on ST RAM whatsoever; imagine video that's based on RAM clipping along at 33 MHz. Now add to that the recent advances in video, specifically higher resolutions, and more colors. As of this writing, the "Chromax" board has a wide selection of video output possibilities, from extremely high-resolution monochrome to "TrueColor" (24-bit color; 8-bits each on RGB), which is industry-maximum as of today. (If you want all the gory intimate details about the SST Expansion Connector, see the chapter about it.)

A Cache?

No, we didn't put a cache on the SST. We thought long and hard about this. You see, we felt that having 4 or 8 megabytes of high speed memory was simply better than having a small 16,000 byte cache memory, in terms of performance per dollar; remember, we wanted power without the price. We took every possible chip off the board we could. That saves you money in the purchase cost.

Besides, there still is a cache in the SST; you're getting the best of both worlds in a way. *Inside* the 68030 chip, there are 256 byte caches for both data and instructions (512

bytes total). Hence, if you get into those small, tight loops, the SST will fly through them in typical cache behavior, only touching memory once. When it "falls out" of the cache, though, is where the SST will shine; it'll burst mode load program/data out of fastRAM at the RAM's top speed, instead of fighting with video in ST RAM, into the 68030's own private caches. This gives high efficiency and is also the reason we can do things like execute in burst mode at 7.2 MIPS.

Well, the board's "design" and "philosophy" was complete; a 68030 for speed, SST RAM to prevent the 68030 from being slowed down by video contention, an FPU for number-crunching use, and an expansion slot. Then we really got down to work.

TT Compatibility

To our complete surprise, long after the SST's specifications were laid out and we were in the process of implementing it, we started learning how close we were to the TT. The TT and the SST share some *very* interesting characteristics!

First, both use the 68030 processor. The TT uses a 32 MHz processor, the SST a 33 MHz processor. (Well, if you ordered that one; we also offer a 16 MHz Option, but you can actually use any speed 68030, rated from 16 MHz to 50 MHz, if you want; you are restricted by your oscillator speed much more than your processor speed.) Realistically, one MHz is not going to matter much either way, as far as maximum speed is concerned.

The TT also has its own fastRAM! It really made me feel good to see the fastRAM board inside the TT. And sure enough, there was a jumper to enable Burst Mode! You see, the reason this made me feel good was that Atari's R&D department had researched the same problems we had and come up with the same conclusion: to get speed, you need memory that video doesn't slow down. They require nybble mode SIMMs to do their burst mode with, though.

The TT does not use a cache other than the 68030's onboard cache, which judging by its performance is plenty. Nor does it use a BLITTER (which is an ST chip which "Blits", or sends, data to video memory very quickly). We, too, consider the BLITTER pretty useless on a 68030 class machine, *especially* with a screen accelerator like QuickST™ or TurboST™ (be sure to get the TT 68030 compatible version).

Once we really dug into the TT, we found out some more stuff. Now, I repeat, we had the SST board underway when the TT showed up. But the *address of the fastRAM was identical!* We'd picked it more or less at random as a handy place to pick up an address select line. All this was starting to get a little bit eerie.

This means that *very likely*, TT specific software will work on the SST, unless it uses the new TT video modes or other hardware that *we do not* support. If a program tries to use those on the SST, it will crash.

TOS 2.05 & 2.06 From Atari

At this point, we needed an Operating System (software) for the board. TOS 1.4 and below *do not work* on the 68030; they were written using shortcuts that work on a 68000, but break on a 68030. TOS 1.6 and above *do work* on a 68030. However, TOS 3.01 and above *cannot* work on the SST; they are TT-specific.

We wanted a TOS that was TT-like, in that it could handle a 68030, but without some of the TT specific features that we didn't have in hardware and which would cause us problems (like 3 serial ports).

Atari came through with TOS 2.05, which is shipped with the Mega ST® machines. It has the "New Desktop", which is very neat, is fully 68030 compatible, understands TT programming, understands fastRAM somewhat, and works well. We were delighted with TOS 2.05 and thank Atari for licensing it to us; without it, we would have had to do even

more major work on TOS to make it work with the 68030.

TOS 2.05 comes on 2 chips, in EPROM (Eraseable Programmable Read Only Memory) form, and is 256,000 (256K) long. (They are already installed on the SST.) Previous Atari TOS's were 192K long, and came either on 2 chips or 6 chips. It is not an easy task to retrofit an earlier Atari with the new TOS; however, the SST takes care of that task. You must, however, *remove your old TOS chips* for the SST to work. That will help your ST to run more reliably anyway, as the TOS chips are a big load on the computer, and removing *any* load is a big help.

TOS 2.06 (which, since we're so lucky, you are getting in your SST) was released *right before* we started shipping the SST. So we decided to delay a little more, and check it out, and, like I implied before, we got lucky. TOS 2.06 fixes some bugs in 2.05 that were causing us problems, and has *built in support for floppy and hard disks!* Anyway, as soon as the final revision of the manual is finished (again), your SST will be knocking on your door.

Editor: You didn't think that all the 2.06 stuff magically appeared in the manual, did you?! Someone had to get it there. I'm up to version 61 so far.

Other Editor: That's nothing. I was up to version 94 on the disk support software; which gets to go into the trash instead of into the package, like your manual does!

520/1040 ST Compatibility

The original SST was, of course, designed for the Mega ST. This is because we had to start somewhere, and the Mega was *designed* to be expanded. There's also lots of room in a Mega (the SST board is about 5 x 5 inches). However, a lot of ST owners have a 520 or 1040, and we have no intention of locking ourselves out of the 1040/520 market. Hence, you can rest assured that 520 and 1040 ST versions are in the pipeline; we are working on that now.

I do want to mention that they're going to be tricky, because frankly 8 megabytes of SIMMs just plain takes up room, and also generates heat. (The 520/1040 *have no fan*, and heat related failures are legion in computerdom). This redesign for the 520/1040 also has to take into account various DRAM boards which may already be installed. So, it's also going to be awhile before it's available. Also, another consideration is that TOS 2.05 or 2.06 *requires* 1 meg of ST RAM in your Atari, and also the ST 520 power supply may not handle even a 4 meg SST.

One technique, if you have a 520 ST and want an SST *right now* is to place the 520 inside an IBM clone case, leave the top off of the 520, and plug the SST in. With a few dollars worth of 64-pin sockets as spacers to clear the other chips, it works just fine, right now. This also gives the heat a chance to escape. If you do this, run backup power and ground lines for the 68000, directly from +5 and GND of the power supply to the 68000's +5 and GND pins. The reason for this is that some 520/1040 motherboards don't really have adequate power supply to the 68000's pins, which is where the SST gets its power from. Then, use an an external keyboard with the various IBM keyboard adaptors on the market; your ST is now in good enough camouflage to smuggle into IBM's headquarters! Incidentally, clone cases also allow easy, easy mounting of hard disks, easy to find power supplies that the SST can't possibly overload (get a 65 watt IBM supply; don't overdo it, or the supply will not be loaded enough to work!), and so on. I currently use two IBM clone cases for my hard disk collection.

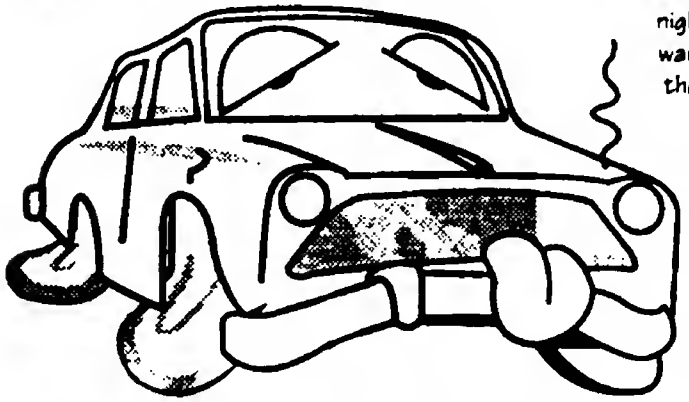
The 1040 ST is trickier. There are *many* different 1040 motherboard layouts. There's even one with the 68000 directly under the keyboard! There's no way the SST will fit there. While this is not a commitment, the logical choice would be a short flexible extender cable to get to the CPU socket, and to let the user install the SST wherever it fits. Anyway, we're working on it.

A Scarred Veteran

That last long Computer Show we went to, the last day of it, in fact, was sure a killer.

The day began with my wife Sandy getting her rental car hit by someone. That meant a couple hours filling out forms for the police. Then, off to the show, late.

Well, ooops. One of our computers had been stolen. Security at these shows can't be that great; the booths are packed too full of people, and computers are too portable and easy to walk off with. There went the demos... another two hours (same police, interestingly!) filling out forms in the incredibly unlikely event the computer was ever seen again (no, it never was) and in the incredibly likely event of having to argue with the insurance company (guess who won). A year later, we still haven't gotten it fully replaced.



And the day had just begun! We passed the day idly improvising demos, and started packing up. We got the remaining equipment back to the place we were staying at and began packing. Sandy stayed up late with the packing tape and boxes, working with a multi-purpose tool (kind of like a Swiss Army Knife) to cut the tape; the tool weighs around two pounds.

Unfortunately, one of our children (who shall remain nameless) chose a tape cutting moment to give Sandy a jolting, enthusiastic good-night hug. The cutting tool in her hand slipped out, and dropped blade-down right into the top of Sandy's bare foot, going in far enough to make you cringe, friend. (I always thought the top of a foot was bone. I was wrong.) Sandy yanked it out and yelled for me, and bled – a multi-tasking wife!

We discussed the options. I said, "Look, it's past 10 PM. All the regular clinics are closed. There would only be the emergency room at the hospital, which we have no idea how to find, nor even how to find out how to find it, and more important, we have no place to put the kids while we get you there somehow." (My brother worked in ER's during his doctor training and told me awful stories of what ER's are like late at night – full of unpleasant people you don't want your kids to meet, police escorting them, and so forth.)

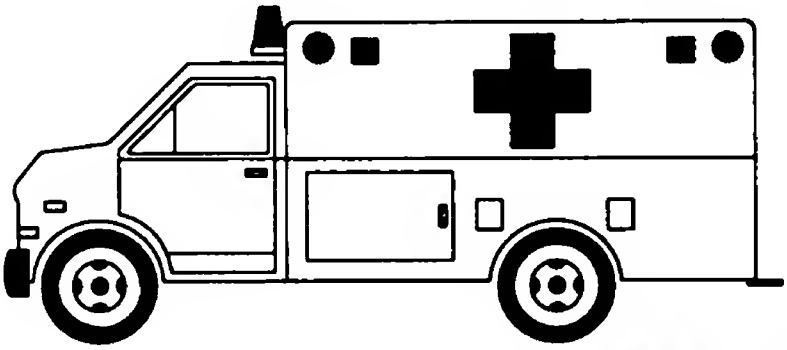
Editor: Not to mention we were in Germany at the time, and knew next to no German.

So, "Why don't we stick a band-aid on this (it had pretty much stopped bleeding already) and go tomorrow morning, and get it cleaned out properly and sewn up?"

Sandy agreed. It will tell you a lot about Sandy that over my protests she went back to packing.

Next day, off to the police station (since we already knew where that was) to ask where to find a clinic. They were even nice enough to escort us poor foreigners there.

A Scarred Veteran



So we finally get in to see the doctor. Amazingly, the doctor refused to sew Sandy's foot up. He explained that it had been more than 8 hours since the puncture, you see. Now, if it had been 7 hours and 59 minutes, he would have cleaned it and sewn it up, but 8 hours... I guess Sandy would turn into a pumpkin or something, like Cinderella's coach. He wouldn't do it. I was completely dumbfounded.

Continuing with the story of the Doctor From Oz, he decided to "immobilize the wound". (Why?) This meant putting Sandy into a **hip-high cast** (No, I am not making this up!) and giving her some crutches to "walk" with. It was one of the older slimy-freezing-cold plaster casts, not the newer high-tech fiberglass models. See you tomorrow, they said.

Well, that was nice. We'd already checked out of the place we were staying at! We went back, explained, and the kind people there let us stay an extra day. You haven't lived until you've spent a day immobilized in a small area with three active children.

Next day, after 1 day of THE CAST, the Doc From Oz removed the cast (!), the whole works. The cut looked... well, **exactly the same**, in fact, like someone dropped a heavy knife into Sandy's foot. The Doc then pronounced himself "finished", and recommended that Sandy stay off the foot. ("It is not fit." Hey, really?!) And off to the cashier we went. (Ouch!)

Luckily, before this little trip/fiasco, we'd stocked up on traveller's medicines, which included some heavy antibiotics. Since the cut hadn't even been **cleaned** at the first visit (after 8 hours, you know... pumpkin syndrome) I was worried about blood poisoning, infection, gangrene, and worse. **Sandy blaming me forever in late-night conversations** for only having one leg. So we started Sandy on a course of antibiotics that would have cured someone of the Black Death.

And, it worked for Sandy. No complications, no infections, no doom, and best of all, only **some** guilt.

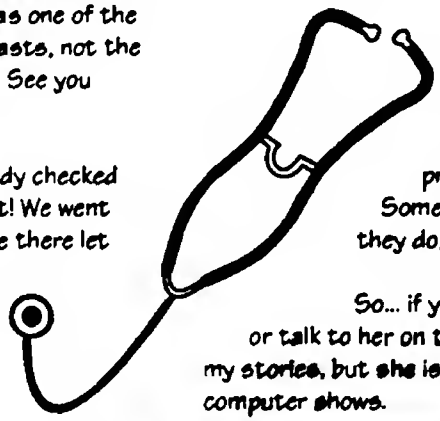
To rub salt into the wounds, so to speak, I talked this over a few days later at a checkup with a doctor in Berlin. He was in a state of complete **disbelief as I described the treatment.**

"They wouldn't clean it and sew it up because it was 8 hours afterwards? What does **that** have to do with it? Sure, they could have done it."

"They **didn't** clean it out? Have they heard of 'blood poisoning'?"

"A **cast!?! Why!?!**"...

...and so forth. He was outraged by the time I was done describing all this.



All in all, I discovered that the medical profession is really a **lot** like the computer profession, or **any** profession, for that matter. Some people are good at what they do, and some aren't.

So... if you meet Sandy at a show, or talk to her on the phone... well, I may have my **stories**, but **she** is the **scarred** veteran of computer shows.

Hard Disks, Floppies, and the SST

SST Disk Compatibility: Required Reading

You absotively possilutely must read this *prior to SST installation*, if you have a disk drive. If you don't, you *should* read it so you know what we're talking about when we mention things like _FRB or somesuch later on in the manual.

As I've mentioned 38 times, (I just love word processors you can do wordcounts with), you cannot directly use SST RAM for either video or disk drives. The disk drive part is straightforward to get around; you use some ST memory to read or write the disk drive, then copy the memory to or from SST RAM.

Where it all gets real interesting, and the reason I've got these dark circles under my eyes, is making certain that various hard disk and floppy disk software ("drivers") is really capable of doing what it is supposed to do, namely talk to SST fastRAM when asked. Putting it mildly, they're all different, and every one of them thinks that ***They Are Doing It Right.***

Of course, the hard part was figuring out *how* each of the different versions of disk software was Doing It Right, and then "fixing" them so it *is* Right, *without breaking it for the ones already fixed!* Talk about your sleepless nights.

Editor: Not to mention that 90% of the work Dave put in turned out to be totally useless with the release of TOS 2.06. (He was only up to version 94.)

Other Editor: Resulting in Minoxodil. And grey hairs. And reburning, retesting, re-editing, and (last but not least) repackaging.

TOS and SST RAM

The first level, of course, is TOS. TOS and your hard disk software are used to working only with ST RAM on a Mega, since that's all you (supposedly) have. However, what happens when you add SST fastRAM, which *cannot* be read to or written from directly?

First off, you're *always* going to have to run a special program called "RAMnbmm" to turn on SST RAM. That's what it's there for; it also does many things "behind the scenes" to help you out. (There's lots more about it later.)

Right now, we can't automatically start SST fastRAM up for you, partially because *we don't know what hard disk driver* you are using. It might need other life support, depending on your driver. Also, once SST RAM is turned on, it *stays* on all session, until you RESET or power off.

Anyway, if your TOS handles all the little "ins and outs" and "gotchas" of dealing with

SST RAM (all in two little chips), your worries are over!

Unfortunately, TOS 2.05 by itself is *not* capable of talking to SST RAM via floppy *or* hard disk. You must give TOS 2.05 "life support", in the form of an "extension" or helper program, for floppy and hard disks before it will consent to talk to SST RAM.

So, with TOS 2.05, you *must* supply floppy disk life support to make your floppies talk to SST RAM. Otherwise, your floppies will talk to ST RAM all day, but you will get spectacular crashes when you try to run anything off floppy into SST RAM.

This life support takes the form of a program I wrote which you either double-click or put in your AUTO folder; either way, it *must* be run once, before you turn on SST RAM, for any session you want to access floppy and the SST. It will last the whole session, and I would recommend you automate the process; it gets *old* having to "turn on" the floppy disks for the 280th time.

Since we include TOS 2.06 with the SST (instead of 2.05), we haven't even bothered to put this particular program on the SST Release Disk; lest someone accidently run it and crash spectacularly.

TOS 2.06 is *very* capable of talking to SST RAM by floppy, all by itself. TOS 2.06 became available literally the same week we were going to ship, so now I am revising the manual for the SST. I have no idea why useful new things always come out when you're ready to start shipping out boards, but it keeps happening. TOS 2.06's floppy SST RAM support is very useful and saves me a great deal of work (which, of course, I've already done for TOS 2.05, and no longer need) in floppy life support programs.

Editor: Isn't that the way it always goes?

If you have TOS 2.06 (and you *will*), you have "no worries" about floppies and the SST; floppy support was built into TOS 2.06. Essentially Atari built into the TOS chips the same ideas as the program I wrote. (No, we didn't trade code.) So, floppy disks are okay "as is" with TOS 2.06. How about hard disks? Well, *they* still need external support.

Hard Disks and SST RAM

Hard disks have *always* worked by loading in their own "driver" (a life support program to make them work), so it isn't asking a lot of the world to have the hard disk "drivers" be made smart enough to deal with SST RAM. The Atari machines, when they power up, just barely touch each hard disk, and if you're a good programmer, you can make that "touch" load in enough programming support so TOS knows a hard disk is there. It is very, very tricky, but it is possible - after all, hard disks have been around for the Atari since 1986.

Technical Note: That "touch" reads in the boot sector; absolute sector 0.

I've spent quite some time testing out hard disk support for SST RAM (which is basically the same as TT RAM).

I haven't been the same since.

I now hold the world's record for crashing an SST by trying to load a program (usually, BOINK, if you want to know) off a hard disk into SST RAM; I'll be happy to face off *anyone*, even at Atari with their TT fast RAM, which is much like SST RAM. Even Ken Badertscher, noted TT gunslinger! I kept count of crashes, and I'm into exponents!

Along the way, we discovered both hardware and software errors, some of them extremely nasty to troubleshoot (be sure to read the Interlude on Heisenberg's Uncertainty Principle). I also discovered that there's this stuff that's supposed to make your hair grow

back; it comes in very useful when I've been tearing said hair out.

If you run a hard disk, and nearly all of you do, then you need *varying* amounts of life support for the hard disks to talk to SST RAM. This ranges from *Absolutely Nothing* (if you happen to be using HDX 4.03 or 5.0, a very popular Atari driver) to *Some Support* (like ICD's 5.anything software, like 5.2 or 5.4.2, or Supra's 3.43), to *The Unknown Territory*, like hard disk makers that are gone from the ST community, or haven't updated their software yet, and ones I can't test because I don't even *have* their hardware and software. With them, even the best life support I can write might not be enough. After all, it is possible to make a hard disk in such a way that I have little chance of helping it to talk to SST RAM; disobey enough "rules", and you've got it made.

It isn't TechnoEngineer level talk to understand how you make hard disks work with SST RAM. So, let's talk.

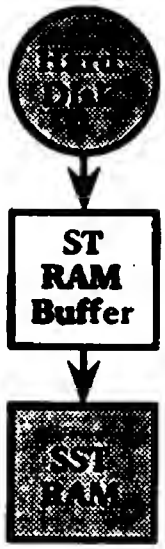
As you'll recall, we hooked up SST RAM so that it had *absolute priority given to the 68030*, for speed reasons. We didn't put in any circuits that could take it away from the 68030, like video or disk.

Now let's say you want to load a program into SST RAM from your hard disk, to run it at high speed. How do you get it there?

Well, ST RAM can still talk to the disks, and always will be able to. So we make an area of ST RAM, called a "buffer", and pull the program off disk into there. Now, it's sitting in ST RAM.

Next, we fast-copy the program from ST RAM up into SST RAM. (Illumination 14.)

Now the program is where you want it, up in SST RAM, just as if SST RAM had disk circuits built in (and cost a lot more). The program starts, runs with blazing speed, your boss is impressed, and you get a raise and stock options. A definite "win".



Illumination 14 - ST RAM Buffer for Hard Disk

Hence, we need several things:

1) An area in ST RAM that the disks can always use,
2) A program that's always around to "help" disks ("life support"),
3) A way to hook this into hard disk programs written by lots of... (ahem) very different... individuals.

Other Editor: Trust me on #3, there's usually always some way.

#2 is a program I call DISKxx.PRG (xx is just a version number, like 94). Its job is handling SST and disk – either writing from SST RAM to disk, or reading from disk into SST RAM. It includes stuff like taking away the disk access from the regular software before it tries to go to SST RAM and crashes, giving the hard disk software an ordinary, dull *ST RAM disk access* that it can handle, and copying to and from SST RAM.

F-117 And The Fabulous Furb

#1 is a little bit of a zinger. Atari calls this area "_FRB", which I pronounce "Furb", or "Fur Buffer" when PyeWackette's in my lap wanting to be scratched, like right now. The "_FRB" is 64K long and *must be in ST RAM*, and is dedicated to chatting with hard disks. You can find its location in the "Cookie Jar", and no, I am not making this up!

Does that mean you can only load a program that's less than 64K long? Fortunately, no; it's possible to chop up one *long* disk access into many shorter 64K disk accesses and have it still accomplish the same thing, and pretty fast, too.

Atari uses the Furb in their TT, because they have the same hassles I do when accessing their "TT RAM" ("Fast RAM", a lot like SST fastRAM) from an ACSI hard disk

(Atari's style hard disk). So Atari developers have known about the Furb for awhile, and indeed, some hard disk manufacturers have updated their software for the TT. (Some have not.)

But me, I ran into problems using the Furb, because every manufacturer does things a *different way*; everyone has *the* clever-fast-copy-loop that happens to screw up my software, so I became a little cautious about the Furb, particularly around crash #6,000. Mind you, it's not a bad idea, and Atari has the Furb down cold perfect as far as I can tell. (We've never had problems with HDX, Atari's hard disk software, in fact; it works in places where it probably shouldn't! Someone there deserves a raise. Her name is Min.)

But I didn't have as much luck with other hard disk drivers, and I traced it down to their Furb conflicting with my software. (I am not saying it was their fault, by the way.) But I decided it was time to hitch up the wagon train, pioneer, and Make My Own Furb.

Other Editor: "... And I'll build you a home in the meadow... "

F117: The Mysterious Stealth Buffer

The F117 Stealth Buffer is our own SST "_FRB."

Editor: "Choke" The "Stealth Buffer"? Oh, DAVID! I married someone who could make a pun like that!?

The Stealth Buffer is identified in system tables by "F117", which happens to also be the number of the Stealth Bomber, you see.

Other Editor: Well... okay. The B-2 is the Stealth Bomber, the flying wing thing. The F-117 is the Stealth Fighter. But would you pass up a pun like that?

The DISKxx program (on your SST Release Disk) is written to intercept and handle *any* disk access that *any* program tries. Whenever we get a disk access, we "chop it up" into 64K pieces, and feed it "through" the Stealth Buffer (a buffer in ST RAM), not through the regular Furb.

For instance, to read 20K from disk to SST RAM, we do this:

- Read 20K from disk into the F117 Stealth Buffer.
- Copy the 20K from the F117 Stealth Buffer to your destination.

Similarly, to write, say, 100K to the disk, we do this:

- Copy the first 64K from your location to the F117 buffer.
- Write the first 64K to disk. (The disk hardware can only write a maximum of 64K at a time, anyway.)
- Copy the remaining and last 36K from your location to the F117 buffer.
- Write the last 36K to disk.

See? Nice and sneaky. What's wonderful about it is we don't *care* what a given hard disk manufacturer has done with *their* Furb. All the hard disk is seeing is ordinary, routine ST disk accesses, which the DISKxx program feeds to it, admittedly all to the same place (the F117 Stealth Buffer), but the software doesn't care. And software that doesn't care is happy software.

When DISKxx powers up... err, is started, it "steals away" the way that people make hard disk requests. Then, it "stands in the way", letting through only disk requests that even old ST software should be able to handle. It seems to work out that way, too; the exceptions I've seen have had to do with 68030 incompatibility or timing loops, *not* disk transfers, which is important. And it does all this *without them knowing about it!*

Sure, "_FRB" is the same thing as "F117", but it is not restricted to the SST, and we can set the rules for F117 *ourselves* - and have already discovered that not everyone is playing by the _FRB rules. We have discovered that we had best keep our hands off the regular _FRB; two of our Beta Testers damaged data on hard disks, and I lost hair before we figured this out!

Technical Note: "F117" is installed in the System Cookie Jar by RAMnbmm.PRG, the SST RAM starter, along with a pointer to a permanently reserved 64K buffer. There will *also* be an "_FRB" cookie; floppy disks require it. Believe it or not, some hard disk software simply doesn't care if an _FRB is out there; it just goes and works (a little like what our DISKxx.PRG does). We don't touch the _FRB anymore, and we don't crash anymore, and we'll leave it at that.

The fastRAM initializer program, RAMnbmm.PRG, is explained in much more detail in a later chapter.

Other Editor: Small's Law #14: If it ain't broke, don't fix it.

Another Technical Note: On the TT computers with TOS 3.01 and up, the _FRB "cookie" is automatically installed early on at startup time. On the TOS 1.x, 2.02, 2.05, or 2.06 series, the _FRB cookie is not installed automatically; if you want it, you have to make it happen.

Hard Disk Drivers & SST RAM

Now let's look at what SST software to use with which Hard Disk Drivers; since you are lucky enough to have TOS 2.06, you don't have to worry about floppy disks at all.

Atari HDX 4.03

Do not use HDX 4.00, 4.01, or 4.02. Get 4.03 and save yourself some trouble indeed. Atari has noted that those are older, phased-out versions of the hard disk software, and that 4.03 is newest in the 4.xx series.

We've encountered a "bit of inconsistency" among hard disk manufacturers regarding use of the _FRB buffer. For instance, Atari's HDX 4.03 (AHDI) does not even seem to need an _FRB buffer; when SST RAM is declared, and the system told about it (see the documentation for the "MaddAlt" command in The Atari Developer Documentation), programs that are SST RAM flagged load into SST RAM with no problem, like magic. It works perfectly. This is really strange, because obviously HDX needs a place to do disk transfers *somewhere* in ST RAM! We found a transfer buffer, but it is *not* declared as an _FRB... so, who knows.

We do have the theory that perhaps HDX 4.03 was "idiot proofed", and if it finds no _FRB Cookie at startup time, then it creates its own private _FRB and uses its own internal area for the buffer, and just never installs it as an _FRB cookie. Theoretically outside programs have no business messing with "your" hard disk _FRB, so perhaps the programmers decided to keep it nice and private. Heck, it was probably a perfectly good idea!

***Caution:** If you put the RAMnbmm.PRG inside your AUTO folder, you must also have it in the main "C" hard disk directory.*

So, if you are using HDX 4.03, all you need to do to initialize SST fastRAM is run the RAMnbmm program. ***Do not run the DISKxx program*** at all; it *will not* cause problems, but the SST will not run as efficiently, since you will be doing two buffer copies each access, instead of just one.

In addition, you can put our RAMnbmm.PRG in the AUTO folder, and the program

which is actually the GEM desktop will load from the TOS ROMs into SST RAM, running faster, and even better, putting the "system supervisor stack" (SSP) into SST memory. This will improve your performance in most programs, as the SSP is among the most intensely used memory in your computer.

Atari HDX 5 or HDX 3

If you have HDX 5, it works exactly the same as HDX 4.03. You just need to run the RAMnbmm program, and do not run the DISKxx program.

HDX 3 is pretty old. While it *can* work if you run both the RAMnbmm.PRG and the DISKxx.PRG, we highly recommend that you update to HDX 4.03 or HDX 5.

ICD 5.2 through 5.4.2

During the development of the SST, ICD's hard disk software went from version 5.2.0 to 5.4.2, where it (currently) is.

With ICD hard disk drivers, run the RAMnbmm.PRG, which will install the F117 Stealth Buffer, *then* run DISKxx to get disk life support. Using the F117 Stealth Buffer seems to stabilize things.

If you want to put the RAMnbmm.PRG file in your AUTO folder, you must *also* have a copy of it on your "C:" directory, so that it can "find itself". Also, make sure that the DISKxx program is in the AUTO folder too.

Since you need to run RAMnbmm, and *then* DISKxx in that order, you will need to remove everything from your AUTO folder, and copy in RAMnbmm, then DISKxx. This will make sure that they are run in the correct order. Next, you can copy the rest of the stuff you want back into your AUTO folder.

Supra 3.43

With Supra hard disk drivers, you need to run the RAMnbmm.PRG, which will install the F117 Stealth Buffer, *then* run DISKxx to get disk life support.

You can run RAMnbmm.PRG file from your AUTO folder, but you must *also* have a copy of it on your "C:" directory, so that it can "find itself". Also, make sure that the DISKxx program is in the AUTO folder too. (See the ICD section for information on doing this correctly.)

Hard Disk Software Compatibility List

So, in summary, here is a handy dandy reference list to show what hard disk software requires which SST software, in order to be comaptible when running with SST fastRAM:

Caution: Incompatible hard disk software can result in machine lockups and crashes, and possible damage to the data on your hard disk.

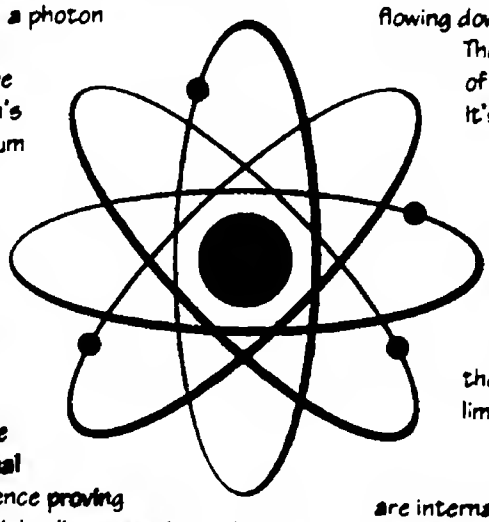
Hard Disk Software	SST Software
Atari HDX 3	Run RAMnbmm.PRG, then DISKxx
Atari HDX 4.03	Run RAMnbmm.PRG only
Atari HDX 5	Run RAMnbmm.PRG only
ICD 5.2 - 5.4.2	Run RAMnbmm.PRG, then DISKxx
Supra 3.xx	Run RAMnbmm.PRG, then DISKxx

We will be updating this list in our newsletter, as we get a wider variety of hard disks and their software tested. Feel free to contact us and contribute your experiences with any different hard disk software which we haven't covered here.

Heisenberg's Uncertainty Principle

Heisenberg's Uncertainty Principle **Or: It's a** miracle these computers even run.

If you're into physics, you've heard of this. Heisenberg's Law states (well, one of them) that you can't **really** know what the heck is going on, because by seeing something, you change it. The common example given is at the atomic level: when you try to observe the atom, you lose, because by having a photon (light) bounce off the atom, you have changed the atom's spin and momentum and direction. You can't really know what the atom was doing before **you stepped in and observed it.**



Developing the SST has been a real interesting experience proving Heisenberg's **Principle** all over again, in the real world. I have grey hairs to prove it.

I guess it must be all the ads for ultra-MHz machines from a hundred manufacturers that make people so blase about the miracle that is occurring.

People, these 40 MHz, 8 megabyte, any brand-name computers are a flat miracle to me. And I know that at each of those 100 manufacturers there are engineers sweating blood to make them work, to give the salespeople something to make brochures about. I know this now because I have been through the process. It is awfully hard. (And NO!, this is not some sort of apology that our board doesn't work right; the SST is tight, fast, and amazingly, **cheap!**)

Let me tell you a little of the miracle, and remember, it applies to any of your 80486/33 MHz IBM boxes or any of the faster machinery you can buy today, not just the SST.

Consider electricity, which is what these circuit boards run on. Think of it as sort of a fluid flowing down the wires, like water down a hose. The pressure is called "volts", the number of gallons per minute is called "amps". It's a reasonable analogy.

Speed

But this electricity runs at a certain speed: **186,232 miles per second** (wow!), or 983,304,960 feet per second. That's the "c" in $E=MC^2$, the famous equation from Einstein: the speed of light itself, the basic "speed limit" of the universe.

And in metric, since the ST and SST are international: **300,000 kilometers per second**, or 300,000,000 meters per second... 3×10^8, a number which I remember well from **my** college classes!, or 26 cm per nanosecond.

That means, in the **ever-so-common** "nanosecond", or one billionth of a second (which you will see in **any advertisement for** computer memory, e.g., "120 nanosecond RAM, 1 **Megabyte**, $30" – which is precisely what your Atari ST uses as relatively slow memory! – anyway, in that one billionth of a second, electricity moves almost exactly one **foot.**

Now look. In a 2 foot square supercheap IBM clone, that means it takes about 2 nanoseconds for electricity whizzing along a wire to get from

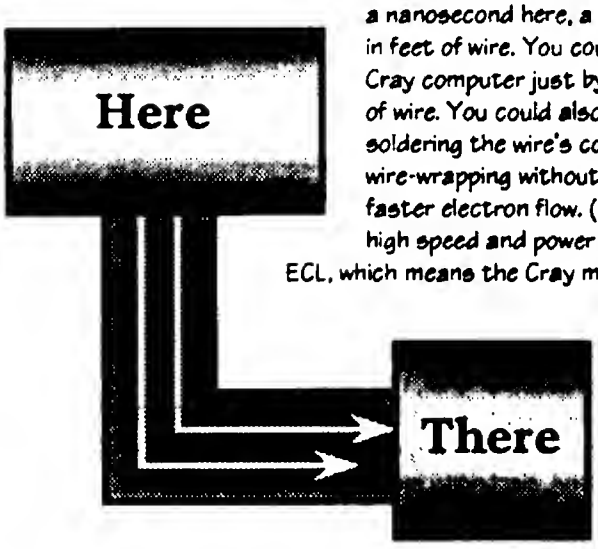
Heisenberg's Uncertainty Principle

one side to the other. And believe me, folks, that is critical. For signals often arrive in a bunch, with their timing related... say, for example, 32 wires of an address, and another wire going "twang" saying, "Okay, the wires at this instant have a valid voltage on them". If the "twang" (we call it a "strobe") wire is run the wrong route on the circuit board (in other words, the curcuit board is laid out badly), you might build in a 2-4 nanosecond delay, and your board will not work, even though it matches the schematic.

Try to find out why your perfectly valid board (according to the schematic) doesn't work. Plug in your instruments. Well, all of them use cables that have electricity moving 1 foot/nanosecond, too! And they add a thing called "capacitance" which makes your signals move less sharply than they once did: instead of springing from zero volts to five volts, they sort of gently ramp up from zero to five. Believe me, ramping up is awful to a computer; the poor thing only understands 0 and 5 volts, and in between is Dead Man's Land, avoided by everyone! (Very often, plugging in an instrument like an oscilloscope will prevent a computer from working, and the engineer has to find a way to get a glimpse of what's going on without upsetting the computer too badly, just to see what's going on.)

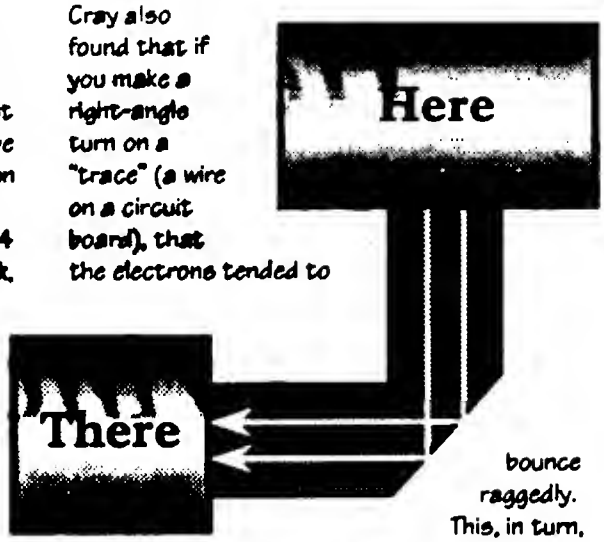
Cray's Wires

When Seymour Cray, the designer of the Cray computers, built his first ultra-fast machines, he used color-coded wires; each different color was one foot (a nanosecond) longer in length. He timed the machine by adding a nanosecond here, a nanosecond there, in feet of wire. You could easily crash a Cray computer just by swapping lengths of wire. You could also reportedly do it by soldering the wire's connections; since wire-wrapping without soldering gives faster electron flow. (He also used ultra high speed and power hungry chips called ECL, which means the Cray machines pull power like an arc welder; the real engineering marvel of the Cray is the cooling system that prevents it from melting into a puddle of molten metal!)



Editor: Ohhhh! Like in Terminator II.

Seymour Cray also found that if you make a right-angle turn on a "trace" (a wire on a circuit board), that the electrons tended to bounce raggedly. This, in turn, made the "wave front" arrive at the next chip raggedly instead of all at once, making for "hiccuping" and crashes. He had to make 45° angle cuts half-through the right angles so the electrons bounced smoothly around the corners, and kept the front edge of the electricity all together, so that all the signals would get from "Here" to "There" at the same time.



I have no idea about how he found this with the instruments available during the Cray-1's design. I believe that what was going wrong (the bug), and the fix, came to him intuitively. That is why he is Cray, and has computers named after him.

Editor: I think debugging is very much like detecting. To quote Sherlock Holmes, "When you have eliminated the impossible, whatever remains, however improbable, must be the truth.".

Other Editor: My reply to Sherlock, "While debugging, nothing is impossible; that's the problem!"

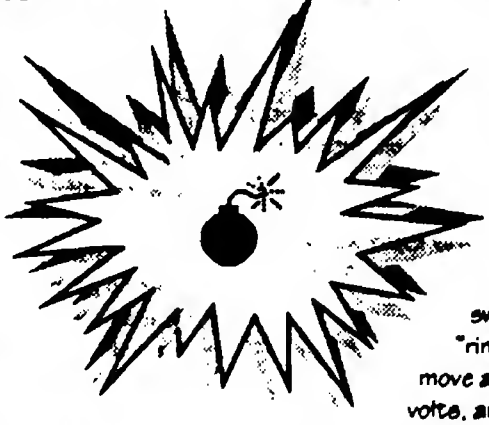
Some of the the chips we are using for the high speed stuff in the SST are rated at 15 nanoseconds to "do their thing". Engineers call it "propagation delay"; it's the time it takes the chip to get the electricity from the "in door" to the "out door". Use a slow chip and you are out of luck. (Even in the now-slow Atari 800 days, I had a project fail because a fast transistor could not

keep up... and that was a 1.79 MHz computer, not a 40 MHz speed demon!)

Faster Than A...

Want a little perspective? Electricity/Light moves around 30 feet in 30 nanoseconds. 30 nanoseconds is the net total time it takes the average modern hydrogen bomb to detonate! One of the reasons they are so helliehly powerful is that much concentrated energy appears that fast in such a small space; for a few seconds, where the bomb was is as hot as the sun, and it gets hot that fast. (But I write this near the end of 1991, and the possibility that we'll see them set off is looking less and less likely as the years go on, thank heavens!)

But... 30 nanoseconds? Heck, some SIMM memory is rated at 60 nanoseconds... and our switching logic, rated at 15 nanoseconds, is **TWICE AS FAST AS A HYDROGEN BOMB.**



I want to sit down right now and say that I feel perfectly justified in plotting bombs on the screen if one of the chips can't keep up! (And may H-bombs move to being just a memory that we can laugh at and make jokes about; I'm not trying for sick humour.)

As you're running your SST, it will be performing at those speeds, day in, day out, the chips thinking it's nothing special, not overstressed or over-run. In fact, if a chip won't make 15 nanoseconds, we have to throw it away!

But at these speeds...

Glitch Chip Corporation

These guys are closely related to the well known Raving Idiot Part Supply, who we ran into with the GCR...

So we were getting ready to ship a bunch of SST's, and we got a notice from the chip manufacturer that some of these specially programmed fast chips were **defective**: if they tried to push too many wires one way, they would generate a "glitch" just two nanoseconds after the switch. And folks, that glitch was plenty to scramble the SST's mind. We had been going out of our minds trying to find the problem on known "good" boards.

It is hard to even **see** such a glitch. If you put a oscilloscope on the line, the capacitance of the scope wire will absorb the glitch, and you won't **see** it on the display.

In fact, the glitch will absorb so very well... **that the SST will start working.** That is a perfect example of Heisenberg's Law: by **observing** the signal, we **changed** it.

Which meant (at that time) that the only way to ship the SST was with a scope. "Just clip on here". ("chuckle") I bet that would have gone over real well with the magazine reviewers!

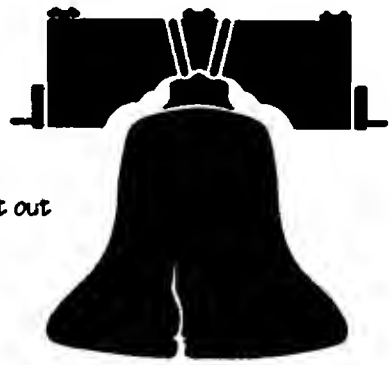
Well, we finally got replacement good chips from another manufacturer (8 - 12 weeks, because they had to redo their design), and had one boring time swapping chips (total drudgery!), and retesting. The SST works now without a scope attached.

And that was just one bug we had to work through getting the SST to you.

Ringing

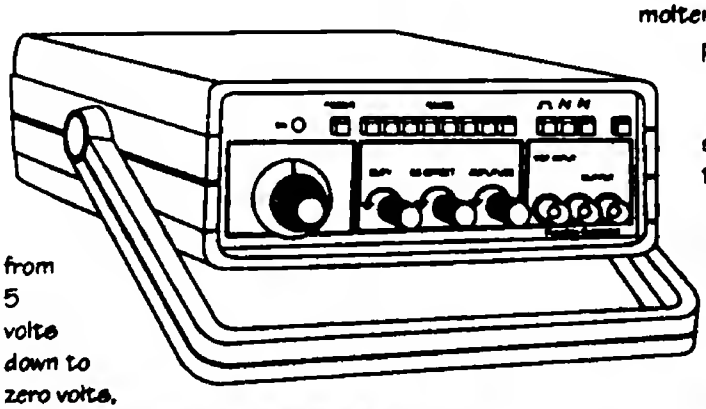
Another charming aspect of electricity being switched this fast is called "ringing". This is where you move a wire from zero volts to 5 volts, and instead of doing it nicely, it goes, "GONG!", and rings like a giant bell being rung. You get a lot of high speed ups back to 5 and downs to 0 and some day it settles down, after completely bewildering everything depending on that wire for timing. ("Make up yer mind, Bud!")

Fortunately, ringing tends to continue when a scope is hooked up, so at least you can **see** it. You can either try to add a dampen-out circuit ("termination") or you can jolt the wire a little less hard, which is what we ended up doing. Honest, you can't just pile in the fastest, hairiest, strongest chips in there; they will ring the whole board right out of working.



Undershoot

Worse, the wire can have "inductance", which is easy to visualize: once you've stopped it, electricity doesn't want to start moving, and once it's moving, it doesn't want to stop – like, say, a train. If you switch



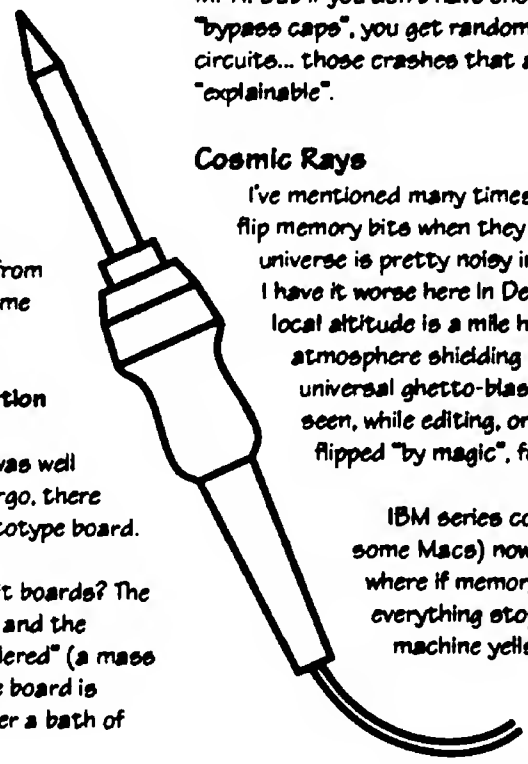
from 5 volts down to zero volts, the electricity just keeps on going down, down, down... in fact, it goes negative, say, -2 volts. This is called "undershoot"; every computer designer knows about it, and it's especially nasty – because negative voltages burn up other chips and destroy them. Memory SIMMs are particularly nasty about undershoot and have long required special pampering. It's become a whole science: how not to blow up other chips on the boards.

Believe me, it is **real** hard to troubleshoot a board that self-destructs in 100 billionths of a second.

Editor: So much for, "This tape will self destruct in 5 seconds, Mr. Phelps.", from Mission Impossible.

Soldering

So I had this hand soldered, prototype SST that was failing every now and then. I clipped the scope on to the power feed and was horrified to see undershoot of negative two volts around the memory chips. I mean, your power feed should be ultra rock solid... one memory component's literature warns you that 0.2 (2 tenths) of a volt variation will prevent the chip from working. I sweated this for quite some time, trying to figure it out.



And then when I took a production board and checked it, there was no undershoot at all, and the ringing was well within .05 volts (5 hundredths). Ergo, there wasn't a problem, just one bad prototype board.

The only difference in the circuit boards? The prototype had been hand soldered, and the production one had been "wave soldered" (a mass production method where the whole board is stuffed with parts, then floated over a bath of molten solder for a few moments, so all the parts are soldered in at once).

I know what you're thinking – screwed up solder job, right? I hate to tell you that the hand solder job was just fine, too... and I even resoldered all the power related joints just to be sure there were no cold solder joints.

I still don't know and, most discouraging, probably never will know what exactly is wrong with that board. That's one reason we test every SST before we ship it.

Bypass

Remember those 15 nanosecond chips? When they do something, changing a line from 0 to 5 or 5 to 0 (and most modern computers only use 0 and 5 volts, to mean "0" and "1" logically)... when they switch, they drain the local electricity from the circuit board, meaning all of it around them for an inch or two. Remember, this electricity is moving at 186,232 miles per second, so we have to worry about inches!

So what you have to do is have, physically close to each little chip, a little storage pond ("capacitor") full of electricity that can be drained into each chip real fast during the switching action. After the switch, the "pond" refills from the main power feed, at 186,232 MPH. But if you don't have enough of these "bypass caps", you get random death in the logic circuits... those crashes that are not "explainable".

Cosmic Rays

I've mentioned many times that cosmic rays flip memory bits when they impact, and our universe is pretty noisy in cosmic radiation. I have it worse here in Denver, since the local altitude is a mile higher; there is less atmosphere shielding me from the universal ghetto-blaster (noise). I have seen, while editing, one character get flipped "by magic", for instance.

IBM series computers (and some Macs) now have a feature where if memory gets a bit-flip, everything stops and the machine yells bloody murder ("Parity Check"), just because of

horrid things like this.

As you can imagine, cosmic rays can affect SST's, too!

And you people at the lower altitudes who think you're safe? Well, cosmic rays (stray radiation) can come out of the ground, too.

And The Future?

In many ways, computer engineers are up against the speed of light/electricity in designing new computers. And that limit is, to our knowledge, unbreakable.

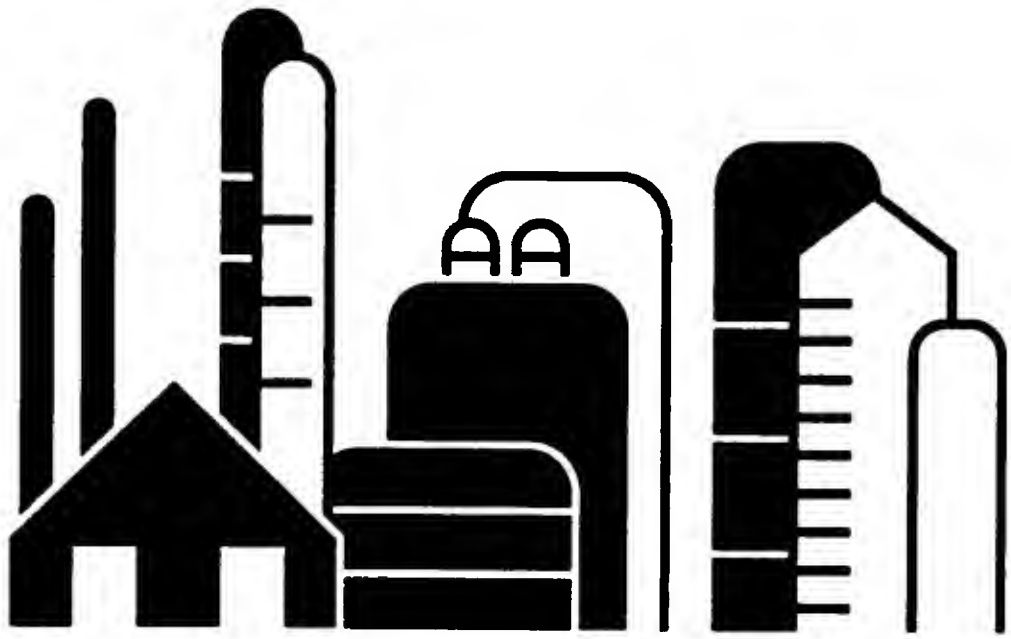
Editor: According to Einstein.

Yet experimental chips are now running at 100 MHz; they generate so much heat that a special chip that cools them must be attached! That's far faster than the 40 Mhz SST.

What's happening is that things are getting smaller. The engineers, faced with the speed of light, are cramming everything into a few 1 inch chips (and internally, the electronics are about a quarter inch square, call it 1 centimeter). And there's work being done on materials that switch faster than our "slow" 15 nanosecond gates; not 70 miles south of where I'm writing this, Seymour Cray is at work on his latest and greatest, the Cray-3, with new materials and new technology.

To me, it say a lot about the human spirit that even when faced with an absolute law of physics - the speed of light - that people like Cray, and thousands of lesser known engineers, keep pushing for more speed, memory, and power.

And that's just the hardware!



Even more Blank "Dedications & Thanks" Page (lest this page be left blank ... Horrors!)

Neil Young for many a late night debugging session's magic; people think I am kidding around when I say things work better while playing "Live Rust" on the CD. The SST and hard disk, for instance.

SST Engine Installation

Installation & Checkout Overview

- Do some "set up" stuff to make sure your hard disks and such will work right with the SST, and make sure you have everything ready for the installation.

- Turn off your Mega and unplug it. Then let it sit for about 10 minutes, in order to let the high voltage capacitors discharge. Take apart your Mega ST. Completely.

- Get your 68000 "socketed". If you already have an accelerator or a PC emulator installed inside your Mega, this is probably already done! Removing the 68000 and installing the socket is 95% of the work. With a socket, you can install (or remove) the SST easily, plug a 68000 in or out, and so forth.

- After installing the socket, plug in the spare 68000 (included with the SST package) to re-test your socketing work and your ST. If everything is okay, remove the 68000.

- Plug as much RAM as you'd like (0, 4 or 8 megs) into the SST. If you ordered your SST with SIMMs directly from us, the SIMMs are already installed. If you're adding your own, 4 megabytes is about right for testing purposes. (By the way, you don't have to set jumpers for memory size or anything like that; it's all automatic.) That's all there is to "configuring" the SST - no resistors to cut or switches to flip when you change things, like in other computers!

- Plug the SST in where the 68000 was.

- Power on, hold breath, and test.

- Relax and Smile. Breathe. It works, doesn't it?

- Test & fine-tune SST RAM, and become comfortable with how it's used.

- Re-install, one at a time, your old D/A's and AUTO folder accessories, to see if any of them won't work with a 68030 processor or SST RAM. Turn 'em off if not; get updates if possible.

- Put the cover back on, put the screws back in, and you're running.

- Test your ST programs, and tune them for maximum performance.

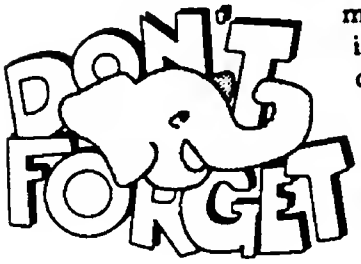
- Invite your friends over and blow their socks off.

Note: A 15 point Karma Penalty will be exacted from anyone comparing SST speed to a TT that someone paid for. (That's 15 lifetimes of dreary peasant-life.)

Before Installing the SST

You will need a soldering iron, solder, a solder sucker, needle-nose pliers, and a pair of sharp-nosed clippers. If you have the "Blitter Fix", you will also need a piece of wire and an X-acto knife.

Make a start-up floppy disk with the DESKTOP.INF file with the BLITTER turned off. Boot your Atari with your Hard Disk off (or unplugged), turn off the BLITTER under the Options Menu, then do "Save Settings" to a floppy. (If your Hard Disk is on, it will save to the hard disk, and you don't want that.) *You may not need this disk*, but it's very difficult to make one after your ST is taken apart! If your BLITTER gives you trouble after installing the SST (and some of them do), then you can shut it down easily with this disk.



Make a backup copy of the original SST Software Disk. Please, oh please, back it up before using it on a new system, which an SST amounts to! Then put the original in a safe place, and use the backup. The SST Release Disk is a standard double-sided disk (9 sectors per track).

Get the *latest* hard disk software from whoever did your hard disk drivers. *You will need this for 68030 and SST RAM compatibility.* Ask them for the "TT compatible" version. Anything upwards of ICD 5.4.2 is fine, and Atari's AHDI 4.03 or 5 is fine, too. Install this on your hard disk, and set it to start up automatically. Supra has not updated their software to be 68030 compatible yet; but we know that version 3.43 works. If you don't update the controller software, *your hard disk will probably not start up* after you've installed the SST. You don't *have* to have a 68030 or TT to update your hard disk software; the new disk software still works with a 68000 machine.

Turn off *all* Desk Accessories and AUTO folder stuff on your hard disk. You don't know which ones are SST compatible; you do not need to be stopped during SST startup by having to manually shut off some D/A that breaks on 68030 processors. (After you've installed the SST, you can turn these back on one at a time, and try them out; remove the ones that prevent a restart.)

A typical way to "turn off" a D/A is to rename it anything but .ACC (like .ACX); for example, ZAPPER.ACC would become ZAPPER.ACX. A typical way to turn off an AUTO folder program is to just leave it in the folder, and rename it anything but .PRG.

AUTOMMU.PRG

Before you take your ST apart, we'd like you to actually setup your hard disk so that it's ready for the SST to start up from. That's why we asked you to go ahead and install "TT" (68030) compatible hard disk software, for instance, and to shut down your Desk Accessories and AUTO folder programs until you're ready to test them for 68030 compatibility, when your SST comes up.

On your SST Release Disk, which by now, *of course*, you have made a backup copy of (do you *really* want to put an original disk into essentially a *new* computer without a backup?!?), you will find the programs AUTOMMUx.PRG and COLDBOOT.PRG. (The x means a revision number, for instance, AUTOMMU1.PRG. Feel free to ignore the revision number; it just helps me keep track of what version of software a person has during technical support.)

AUTOMMU is an AUTO folder program that prevents a problem in the 68030 processor when it is attached to Atari's ST hardware. We ran into this bug early on, but it was very intermittent and hard to pin down, one of those once-a-day things. We finally

had to lock hardware into a test mode and let it run for hours before it "glitched". Once we'd traced it out, it became clear what was going on, and that something was needed to prevent the possibility of the problem surfacing in your SST; in fact, I don't think it is possible to fix this problem in hardware, short of modifying the ST.

It is ***ABSOLUTELY ESSENTIAL*** that you use this program!

If you do not, the ST with the SST installed then has the option of running wild every now and then and doing very, *very* bad things to your disks. We know; our Beta Testers trashed a couple of hard disks and many floppy disks before we figured this out. (Thanks, guys! They do, however, have some of the best backup/restore tools around, from *long* experience Beta Testing.) Other really strange things happened with the ST hardware because of this problem, too; it is impossible to predict just what effect will happen or when, but I can reassure you that you'll know about it "soon enough".

AUTOMMU is a fairly short program, but it is enough to cure the problem for your entire ST/SST session; it sets up hardware which prevents the 68030 from going along the path where it ends up in trouble. Other than that, AUTOMMU is pretty dull. AUTOMMU has no side effects, won't affect anything, does not cause compatibility problems with other programs, drinks cheap beer, and the only sign you'll have that it has run is a short message saying its name, to let you know it's there upon startup. AUTOMMU can run in any "position" or order in the AUTO folder, *as long as it always gets run*, and it only takes a very short time to run.

AUTOMMU.PRG MUST be in your AUTO folder before you attempt to start the SST for the first time!

I don't want to scare you into thinking you *will* see problems; it has been many months since anyone testing an SST has seen this glitch! Our solution works well and doesn't take *any* chances with the SST, or with our hard disks. However, I do want to be absolutely clear on this point; many parts of the SST are optional and user-installable. ***This is not optional***; to get *reliable* SST operation, you *must* have AUTOMMU in your AUTO folder.

The reason AUTOMMU works so well is that it *is* located in the AUTO folder. As it turns out, the ST/SST interface can't get into trouble from the time you power on until well past running all the programs in the AUTO folder! Hence, the SST will get the ST to the AUTO folder; AUTOMMU will handle it from there.

If you want the gory details, when we researched this problem, we found out that until the moment that the GEM "Desktop" comes up on screen (with windows and icons and such), you don't get this problem. After GEM "Desktop" comes up, though, look out! So we made an AUTO folder program for you to fix this. Why? Because AUTO folder programs are *always* run before the GEM Desktop starts up. Hence, if you put AUTOMMU into your AUTO folder and leave it there, you will have no worries. There is never a time when you are not protected, since you are running it every time you boot.

If you use floppy disks, simply put AUTOMMU.PRG into a "startup floppy" with an AUTO folder.

If something unusual happens and AUTOMMU.PRG isn't run, you can *always* run it manually; just double-click it. It can also be re-run multiple times without a hitch, but that won't gain you anything.

For you technical people, so you know, the program that switches on SST RAM, RAMnbmm.PRG (which we'll get into later), takes over from AUTOMMU and incorporates AUTOMMU's functions inside it. But, in order to get to the point during bootup where you can *run* the SST RAM program, you need AUTOMMU tranquilizing the 68030!

COLDBOOT.PRG

COLDBOOT is a less drastically important important program than AUTOMMU, but is still a darned handy thing to have around. I've written various versions of it over the years; this is the one I now use. COLDBOOT.PRG also goes into your AUTO folder.

The problem COLDBOOT handles has been around for a long time: Warm vs. Cold "RESET"s. As it turns out, when you press the keyboard RESET keys (Control - Alt - Delete or Shift - Control - Alt - Delete), or if you press the Atari RESET button, you don't always get the same sort of RESET! There are two kinds: a "Cold Start", which is the same thing as turning the Atari ON, which forces everything to be new, and a "Warm Start", which only sets up certain portions of the ST and its operating system, and *assumes* all is well with other portions (which is not always the case).

The keypress Shift - Control - Alt - Delete tells the Atari ST to do a Cold Start, and Control - Alt - Delete does a Warm Start. The RESET button in the back of the ST sometimes does a Cold Start, and sometimes a Warm Start. The problem comes when you get software problems or a glitch in some area that a Warm Start doesn't fix up. You can press the RESET button all day and it won't do you any good! (Power OFF and ON is the only solution.)

Note: As of the first SST software release, Control - Alt - Delete works correctly *only* if you put COLDBOOT into your AUTO folder. Shift - Control - Alt - Delete crashes the system; it causes all sorts of problems. So, to reset with a keypress, use Control - Alt - Delete with COLDBOOT, and you will get a Cold Start.

How does the ST (or SST) tell the difference between a warm and cold start? After a successful Cold Start, a few memory locations are written with very specific numbers, that mean, "We Cold Started OK". When a RESET happens, the ST checks those locations. (The idea is, these locations will be zapped if the power has been turned OFF.) If those locations still have the right numbers, it does a Warm Start, and part of the startup is skipped. (The odds of memory having exactly those values by chance are extremely low). If those values have changed, even by 1, then the Atari does a full, complete start. This includes heartwarming things like clearing out memory tables and setting them up from scratch, *which is what you want.*

A very good analogy is that a Warm Start is gathering up most of the cards into a pile; a Cold Start is buying a new deck and shuffling them a few times!

Editor: David, see if I play BlackJack with you again! No wonder I never get any aces!

This can be tricky, because memory has amazing persistence in the Atari, even with leaky DRAMs. If you flip the power switch OFF, then back ON again a second or two later, you will very often find that memory has not "faded away" from lack of power, as you would expect! The Atari checks its "Cold Started OK?" memory locations, *they are still set right*, and you Warm Start. This is really awful, because typically memory really is fouled up by the power off/on transition, which no computer should be expected to ride out untouched, and the Atari can do a spectacular crash/belly flop when it relies on memory that has leaked away.

Now this is actually good news in terms of your ST's ability to handle power "glitches" and "spikes" - in spite of everything, your ST's memory gives a darn good try at hanging in there! - but bad news in terms of RESET. Ever flipped the power off and on fast, and have your ST just sit there? It probably crashed on a Warm Start.

To make truly *sure* you get a full RESET, you ***must*** power OFF, count out ten seconds to make sure memory all fades away from lack of power, then power ON. Only then, when the Atari checks its "Cold Started OK?" memory locations, will it find them changed

from lack of power, and do a real Cold Start.

Buying a brand-new ST and plugging it in for the first time is also a *pretty* good way to achieve a Cold Start.

Now, I am not speaking of *physical* damage to your ST or SST's memory! It is all just damage to what is written there, and can all be cured.

Editor: You'd know, wouldn't you, Dave? See the interludes about computer shows, Destruction of ST's in Your Lab, etc.

Other Editor: MY Lab? What about the cat? She owns the place.

Editor: Back to the manual, Dave. Worry about the cat hair in your computer later.

On the SST, Warm Starts like this can get really old really fast. The 68030 can do marvellous things if you set up some tables and tell it where they are. We set up some rather complex memory structures, and assume that you are coming up off of a Cold Start when we do so. If you're Warm Starting, you can end up with twin copies of some of these structures, multiple AUTOMMU's, filled up and overflowed system data structures (because they are loaded twice), and so forth. I know; *I did it*; I then wrote COLDBOOT after much wailing and gnashing of teeth.

As a quick example, you're going to find out shortly about using the program RAMnbmm to switch on the SST's extra memory. If you run that program twice, *you will crash*, because it assumes in many places it is being run off a Cold Started machine. (For instance, it tests and totally clears out SST memory; that's going to be pretty hard on the special things that get automatically stored in SST memory!) Probably the best I can do is make the program unable to run twice without a Cold Start "nicely", by giving you an error message. On the other hand, a crash is the ultimate error message.

If you put COLDBOOT.PRG in your AUTO folder (or just plain run it), then the next time you RESET your Atari (either with the keypresses or the RESET button or with a quickie power off), you *will* do a Cold Start. COLDBOOT goes through and systematically zaps all of the "Cold Started OK?" memory locations, making them look like a major power off happened; when this happens, your ST absolutely *will* decide to Cold Start when you RESET. And various rather strange bugs caused by Warm Starts will quit happening.

Running COLDBOOT will not, by itself, restart your Atari. It just forces the next RESET to be a "real" RESET, which is a safe, known starting position; you really don't know for sure where you're coming from on a Warm Start. I got sidetracked by several bugs caused by a Warm Start while working on the SST, so I'm passing along COLDBOOT and this advice; I'm not the only programmer that wonders (and has to figure out) what their programs are doing when given a machine in an unknown state.

There are some programs, which you can identify because they call themselves "RESET Proof", which could *possibly* undo what COLDBOOT does and set you back to a Warm Start. (I tried to make that hard for them to do, by the way!) One class of these programs are "Reset-Proof RAMDISKS", which I have a bit of advice about.

RESET-Proof RAMDISK?

I'll tell you something that not everyone is going to agree with. There are programs called "RESET-Proof RAMDISKs". These are high speed "disks in RAM" (a disk drive faked by using memory, which is very, very fast) that are said to survive a RESET. When you check them out, they do indeed seem to survive a RESET pretty well.

The problem is, I found out, *completely by accident*, that when you RESET the Atari, that every now and then (definitely not every time, but definitely some of the time),

memory gets "shotgunned". That's a good word for it; bits are randomly flipped throughout all of memory, including the files stored in that RESET-proofed RAMDISK. Hence, your files can get a little bit of damage, and you might *never know* until you find out the hard way. Many programs will run just fine with a bit or two flipped until you happen to hit on the affected section; data files will work just fine, and so forth. The *only* way to catch this sort of thing that I am aware of is CRC checking, which almost no one does. I am telling you this not to irritate the programmers that have written these RAMDISKs; I am telling you because *I lost a lot of work* the hard way.

If I needed to, I could probably track down the precise hardware snag that can cause it; my guess would be that the RESET arrives at the exact *wrong* time, as memory is being switched between the CPU and the SHIFTER, and a RESET is a pretty strong dose of medicine for the ST, a lot like kicking it. I also know I've talked to a good hardware engineer who told me about how this could happen - a RESET causing one ST chip to ripple through memory, flipping things around, because chips go a little weird during RESET; it is an undefined time. As I said, ST memory is pretty tough, even to surviving (mostly) a power off/on, but, you can't expect it to keep juggling memory and so forth when you turn off its video fetch (which a RESET does) *which does memory refresh*, for awhile. (We talked about that early on in the manual; now you know why we put that part in!). Dynamic RAM (DRAM) is just not that stable, and needs all the help, and distrust, you can give it.

If you'd like to check this out yourself, try flipping the power off and on with a video screen up on your Atari. I think you will find that the screen "comes back" briefly with increasingly more black and white dots sprinkled randomly through it; that's the "shotgun effect" I'm talking about. When one of those sprinkles hits the "Cold Started?" memory locations, which is pretty random, you will finally Cold Start, and get a cleared video.

The same thing will happen with your RESET button; it will just take longer for the random bit-flips to show up.

I've repeated the test across many ST's, and it isn't just my machines. While I know that RAMDISKs are a great thing, I just can't recommend anymore that you use the RESET-proof variety. (Fortunately, most RAMDISKs don't *require* the RESET-proof option; it is only an option.) I do recommend a RAMDISK to many people, particularly single floppy drive users; I just have been burned by *my* experience on "RESET-proof".

Okay, presumably your hard disk is all ready now; prepared for its first taste of the blazingly fast SST. Onwards!

Static Electricity

Now we need to talk about static electricity, which can be *very bad news* for your computer and your SST.

Every object that's electrically conductive, like the Mega, the soldering iron, and you (with all that salty water in your veins, you are very conductive!) has a certain amount of static electricity voltage in it. When two objects touch, they equalize the static electricity between them; if one object has 20 volts, and the other has 0 volts, and they touch, they both end up at 10 volts. There is a definite current flow the instant they touch.

Walking across my rug with shoes on, I can build up a static potential of more than 20,000 volts easily. I know, because that's enough voltage to cause a spark to jump through air, which is not very conductive, and I can get a good 1 inch (2 cm) spark from my finger to a light switch or any other ground. In fact, I can make a fluorescent light bulb flicker in my hand by just holding it and walking around; now *that* is static!

The commonest way to blow a chip is to hand it to someone; the moment both your hands touch the chip, your bodies' electrical charges equalize *through* the chip. Snap,

Crackle, Pop, it blows. The way to avoid this is touch whomever you're handing the chip to first, to equalize static, then hand the chip across - or, safer, to drop the chip into their hand, so it only contacts one person at a time.

What we're trying to avoid is you getting static electricity built up in you, then discharging it through a part of the ST or SST, blowing out that part, by touching it.

After you've opened your Mega, look at your ST circuit board (also called a motherboard). There's shielding solder-pads on the very outside of the circuit board, all around. If you touch one, it equalizes your body and the ST's static voltages. Now, if you touch a chip or something, you're safe. *If you get up and walk around, touch that ground pad again to equalize yourself to the ST.*

Static Electricity Precautions

Don't work on a known static surface (a rug is bad). Taking off your shoes is a very good idea, actually; shoes generate a lot of static. In winter, when the air is drier, static is usually far worse than in summertime. Humid areas have less of a problem with static than dry areas; where we live (Colorado) is quite dry.

Ok, now you know how to static-proof yourself. Please be aware of static, and take static precautions, throughout the *entire* installation procedure.

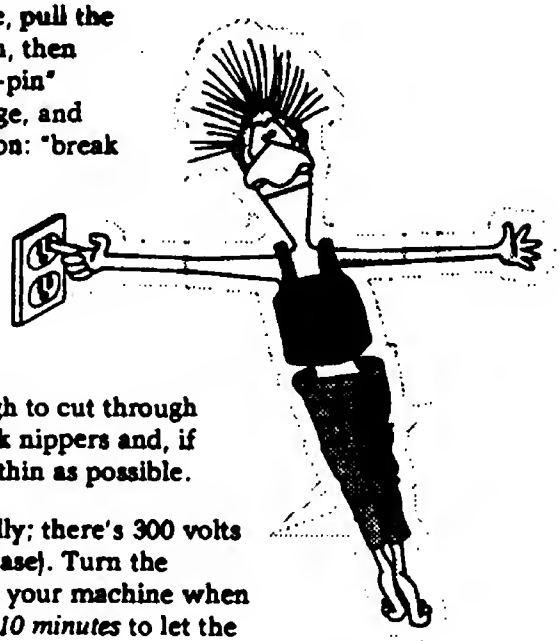
Socketing the 68000

Actually, installing the SST into a Mega is about as easy as installing any accelerator. Laying the groundwork (getting the 68000 socketed) is 95% of the work.

If you don't have your 68000 socketed, you need to do so. This is *not* the place to learn desoldering; I killed a 520 ST doing a 68000 desolder, and it's not fixed yet, over a year later. I've desoldered lots of stuff, too. If you're not fairly comfortable with soldering and desoldering, *take your machine to someone who is.* In a later chapter is a list of Authorized SST Installers.

If you're determined to do this step yourself, here's a quick overview:

You clip off all the 68000's pins as close to the chip as possible, pull the remains of the pins one at a time with pliers and the soldering iron, then desolder the holes, then cleanup, then install a quality, "machined-pin" socket. Do not try to save the 68000; a 68000 is in your SST package, and you will almost certainly lift a pad or zap a trace trying. (Translation: "break your ST".) I did; that's why I'm passing on the advice. (Anyone want a dead 520 ST?)



In Preparation

- Get a pair of very fine-nosed wire-cutters. In the USA, I've found that Radio Shack's "nippers" or Sear's "Craftsman" toolkit nippers are excellent. They need to have jaws small enough to fit around a 68000 pin from the top, and be sharp enough to cut through the pin from the top. For the best possible tool, get the Radio Shack nippers and, if necessary, file the tips from the outside, *not* the cutting side, to as thin as possible.

- ***Ensure* you are safe from your power supply.** It is deadly; there's 300 volts (not 110 or 220) waiting to knock you around (and that's the best case). Turn the machine off, unplug it for safety's sakes (what if your kid gets into your machine when you're not around?), and then *let your unplugged ST sit for at least 10 minutes* to let the supply voltages bleed down. They don't go down in a couple of seconds.

(I found out the hard way; it threw me across the room, and that is considered "lucky". It could have stopped my heart!) Unplug it, and let your Mega sit awhile.

➟ Clear off an area about a meter (yard) square, so you have a place to put all the miscellaneous bits and pieces when you take apart your Mega. Also make sure you have a good light source, so you can see what you're doing.

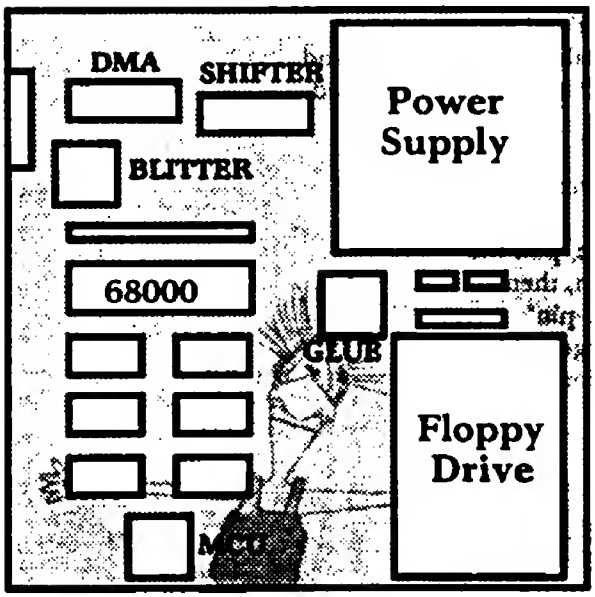
Dismantling Your Mega ST

➟ Take apart your Mega. This consists of flipping your Mega upside-down, removing every screw in sight on the bottom, including the one under the label saying, "If you remove this label, your warranty is history." (This is one of the reasons we highly recommend having your Atari Dealer do the socketing.) Please keep track of what screw goes where; there are different length screws. (I use colored felt-tip pens, and mark the top of the screw, and the plastic hole it goes into, then match colors to put everything back together.) Then, the top lid and the disk drive, and possibly the power supply, will be ready to fall out. Turn your Mega back right-side up; the screws will drop out - catch them!

➟ Remove the top lid. Note the wire leading to the battery compartment; unplug it. If you've had your Mega apart before, you know about the shielding inside.

➟ Remove the shield. Straighten the numerous little tabs that are part of the underside shield that hold the topside shield down. I have seen a few Megas where the shield is soldered down as well; you will need a *heavy wattage* soldering iron/gun to remove this, as the shield makes a fine heat-sink.

The disk drive will fight with you over the shield; you have to get the portion of the shield that goes under the drive out. This is why it's useful to remove the drive mounting screws, which hopefully you've already done. (Illumination 15 is a block diagram of the Mega motherboard.)



Illumination 15 - The Mega ST Circuit Board

➟ Once you have the shield off, unplug the disk drive and remove it. It has two connections. One is a 34-pin ribbon cable that simply pulls off. It can't be re-installed incorrectly. The other is a 4-pin power cable which could be re-installed incorrectly, and burn out the drive, if you tried hard. I think this is something you want to avoid. So, take a moment and mark the power cable's orientation. The easiest way I've ever found to do this, on floppy disks, hard disks, tape drives, and many other devices, is to use a "Sharpie" permanent marking pen, and mark both one side of the power connector and some portion of the frame of the floppy drive right next to the connector (ideally, only one line is drawn for both, so it's obvious which way everything goes).

The floppy power connector may need to be bent upwards or downwards slightly to be removed; it has a tab that helps hold the connector on. You need to slide over it. I can't tell you the direction as it varies with different drives, and Atari has used Epson, Mitsumi, Chinon, Brand X, Brand Y, and Brand Z drives in their machines; I don't know which you have.

➟ Next, remove the power supply. By now, if you've read my fifteen warnings about it and are still working with a live power supply, I figure there's no hope of warning you; I'll just tell you it was *three days* before the muscles in my back stopped hurting from where the power supply that zapped me contracted them (I suspect I either strained or tore a muscle in there).

The power supply unplugs from the circuit board with a very obvious connector located between the power supply and the floppy disk drive (which is gone by now). Then, the power supply more or less "lifts out". This is a little tricky. The power supply has parts that extend out of the plastic case that tend to hold it down, so you have to sometimes tug

a bit to get it out. Look at the openings in the plastic case for the power switch and the power plug and you'll see the problem; maneuver the supply by those, and you'll have it out in no time at all. There *may also* be screws holding down the power supply; if so, remove them.

Removing the power supply and floppy drive are necessary because we have take the ST circuit board out of the Mega case, in order to desolder the 68000. You don't want these parts falling off and crashing to the table (which could ruin the floppy drive, in particular) when you flip the ST motherboard over.

Next, pull the circuit board towards the front, out of the case; you have to make sure the "RESET" button clears the case. Next, you need to remove the metal shielding on the bottom of the circuit board.

Assuming the Mega is on your workspace with its front facing you, its floppy disk drive area will be to the right-front, the power supply area in the right-back (careful!!), the TOS ROMs in left-front, 68000 (or the socket) in left-center, and various chips (Blitter, etc.) behind the 68000. The 68000 is about 3 inches (8 cm) long and about 1 inch wide (2.2 cm), and has 64 pins. (Illumination 15, previous page.)

Remove the BLITTER Modification

On some Megas, a BLITTER chip was used that needed help to work. This "help" was hand-wired onto the circuit board in the form of a 7474 chip, usually glued on top of the 68000. (If you're curious, the 7474 kicks all the other chips out of memory while the BLITTER is accessing memory, to prevent collisions and subsequent insurance claims.)

If you don't have this modification, then skip on to "Remove the 68000".

If you *do* have this chip, remove it. An equivalent chip is on the SST board (not a 7474, but it does the same thing). You'll want to clip the wires going to the ST board and ditch the chip. It might be a good idea to leave 1/2" of insulated wire left of the wires going to the chip to mark where the wires came from in the event you ever decide to permanently re-install this chip.

Now you need to fix the cut trace, found on the underside of the ST board, which you can't do until you've removed the 68000.

After you remove the 68000, if you look carefully (and you must), on the underside of the ST motherboard, right where the 68000 is soldered in, you will see where someone cut one trace as part of the 7474 installation. You must fix this. It takes a *short* piece of wire. There's two easy ways; either find where the trace goes, both ends, and run a wire between them, or (easier), *gently* scrape the coating from on top of the traces for 1/2" (1 cm) on either side of the cut, until shiny metal shows, and solder a bridge/jumper wire across the cut. Don't try to just bridge it with solder; you'll end up with cracks later on if you do it that way. Give the solder something to hold on to.

If this seems like not much fun, imagine being the person whose job it was to install 7474's into Megas by the hundreds!

Question from the crowd: If I have to re-install my 68000, won't I need this 7474 chip back in to make it all work?

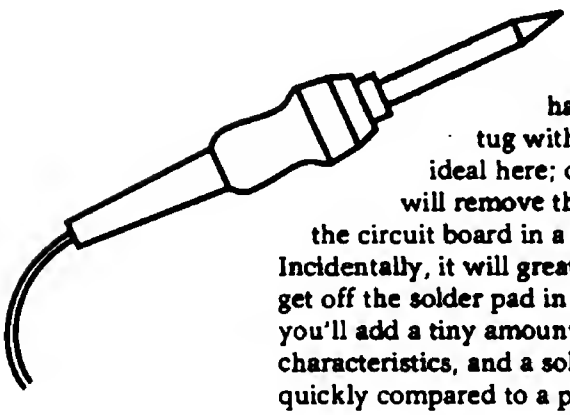
Answer: No, if you de-select the BLITTER from the desktop. You need a start-up floppy disk with the "Save Settings" DESKTOP.INF file with the BLITTER turned OFF, and then the ST won't try to use the BLITTER. You should have already made this disk, because it's a little late to make it now.

Remove the 68000

➥ You have 64 pins to cut. You want to cut them as close to the 68000 as possible, on the horizontal portion of the pin (not the vertical portion if you can possibly help it). Come straight down on each pin, with the clippers held vertically, and cut the pin. (Refer to Illumination 14 if you don't know which chip is the 68000.)

Again, please *do not* try to "save" the old 68000. *We include a new one with the SST*; they cost $7 dollars; and most importantly, trying to save the old 68000 will almost certainly give you circuit board damage. The way those 68000's are installed makes them extremely difficult to de-solder without lifting up a solder pad, damaging a connection and so forth. Like I mentioned once or twice before, I have a 520 ST I killed trying to save the 68000 to teach me, and I have done nearly everything possible to try to restore it (continuity checks, short circuit checks, logic analyzer, in-circuit-emulator... and I simply do not know what is wrong.) After you get the 64th pin cut, the 68000 will be ready to lift out.

Something fun: Take a file, and file down the nubs that are left of the pins. Next, drill a hole through the 68000 on one end. Instant nifty keychain! ("Is that a 68000?") Believe me, it's an instant conversation piece.



➥ Okay, now plug in your soldering iron and get your solder-sucker. What you want to do is heat the pin and (hopefully) suck it out with the solder-sucker. Otherwise, you'll have to heat the backside (bottom) of the circuit board, and gently tug with needle-nosed pliers from the top of the board (two people are ideal here; one to hold the soldering iron, and one to pull the pins out). This will remove the pin in that one solder pad. I would really recommend putting the circuit board in a soft vise for this, and using two people - one to tug, one to heat. Incidentally, it will greatly help heat transfer of the soldering iron to the board (and let you get off the solder pad in far less time, which is far less potentially damaging to the board) If you'll add a tiny amount of rosin-core solder on each pin; solder has excellent heat transfer characteristics, and a soldering iron with a bit of solder on it will heat a solder pad very quickly compared to a plain soldering iron without solder on it.

Once you've removed all the 68000's pins, you then heat and remove the rest of the solder (using the solder-sucker tool) from the solder-pads on the backside of the board. When you're done, do the usual cleanup of any splattered solder or tiny solder connections that happened between traces.

➥ If your ST had the BLITTER modification (if it didn't, skip this part), now is the time to fix the trace that was cut whenever the 7474 chip was installed. I have seen Megas where this trace was cut on the top, and others where it was cut on the bottom; fortunately, the cut is reasonably obvious. It looks exactly like someone took an X-acto™ knife to the board and slit it right through the trace, or a drilled a hole through the trace.

Again, either find the ends of the trace, and solder a wire between them, or carefully scrape off the solder-mask coating from the trace itself, and solder a bridge across the actual cut. Don't try to "get by" with just solder; that will give you nothing but trouble later on. Use a small bit of wire to support the solder, and you'll have no trouble soldering directly to the trace.

Make certain you've cleaned up any stray solder that splattered during desoldering on both the top and bottom of the board; a really strong light is a big help here.

Install the New 64 pin Socket

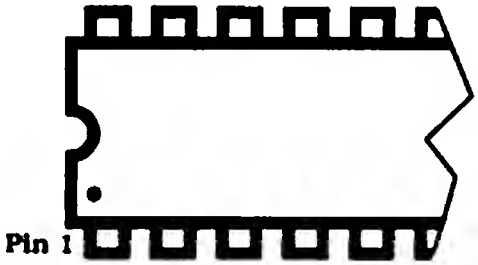
Next, install a machined-pin socket (like that one in the SST package); pin #1 is the end with the half-moon indentation, and should be the end farthest away from the floppy disk

drive. (Illumination 16.) I usually hold the socket in, solder one pin to hold the socket down, then do the rest, to prevent the socket falling out while you turn the board upside down to solder it in. Take your time and you won't have any problems.

Note the "machined-pin" socket; you do not want a "solder-tail" socket. Solder-tail socket pins are thin and rectangular; machined pins are circular. That's the quickest way to tell them apart; the other quick way is which costs more (machined). Well, you get what you pay for.

Again, make certain you've cleaned up any stray solder that may have splattered during soldering on both the top and bottom of the board.

Okay, your Mega ST now has a socket where the 68000 was.



Illumination 16 - Pin 1 Orientation

Reassemble and Test Your Mega

Why test? Well, you don't want to have a flaky machine and not know if it's caused by the SST or the Atari!

So, install the new 68000 that came with your SST. Pin #1 of the 68000 is the one with the dot by it, on the end with the half-moon shaped dimple; it needs to be matched up with pin #1 on the circuit board, which is the end *farthest from* where the floppy drive goes. (Illumination 16.) Start the pins on one side first, then the other. Now, push firmly, watching to make sure none of the pins bends as you push the 68000 in.

Next, re-install the bottom shield, ST circuit board, power supply, and floppy drive, so you can test out your machine. Make sure you reconnect all the cables properly, and double-check them *before* you turn the power on! (Be sure to keep your fingers out of the power supply.)

This test will make sure the socketing is solid, and that a trace didn't get overheated and break during the socketing operation. Let the ST get nice and warm during testing; that tests it better (in particular, bad "cold solder joints" only show up as bad when warm).

Remember that if your Mega had the BLITTER modification, you will need to start from that floppy you made before you started the installation.

Okay, now you've got a solid, working Mega ST with a socketed 68000 (or accelerated 68000; that's fine to test with, too, but more expensive to replace if something is shorted).

Next, turn off your Mega and unplug it for at least 10 minutes, to let the power supply capacitors discharge again.

Using a flatblade screwdriver, knife, or whatever, lift out the 68000. Do it a little at a time from each side, or you'll bend the pins on the opposing side. **Also, be *very careful not to dig into the circuit board underneath the socket* with your screwdriver**, or you could end up repairing traces! Take it easy and it'll be out in a little time. Put the 68000 back into the black conductive foam. If you don't have enough of that, put it in aluminum foil; that's safe, too. Stick its pins through the foil so they all touch. The idea here is to prevent any imbalance across the 68000's pins; if they're all hooked together through conductive foam or aluminum foil, it can't happen.

I have never blown a 68000 from static electricity, and I've been pretty careless sometimes, and we have an all-time static generator in the carpeting in the Gadgets office, not to mention PyeWackette. (Don't sweat this too hard; a little caution can go a long way.)

We have, however, blown a 68030 from static, so don't push it too hard... of course, it was an expensive, 50 MHz one.

Editor: Naturally. It must have been a Monday.

Remove the TOS ROMs

In front of the 68000 are the TOS ROM chips (Read Only Memory), either 2 or 6 of them. These ROMs contain the built-in programming for your Mega, from TOS 1.0 (original) to TOS 1.4 (most recent) to possibly other TOS's. We have to remove them because they *will conflict* with the TOS in the SST board. (In specific, in locations 0-7, for you programming gurus, which holds the startup location. If you have two TOS's in there, big trouble.) It will also reduce the capacitive load on the ST's system bus and make it cleaner, and that's one touchy, weak area; this will help!

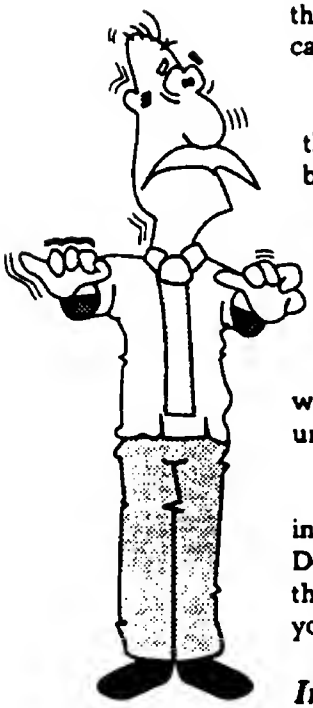
Label them so you'll know which one is which, and draw a little sketch of their locations. This is so if you need to reinstall the old TOS, you can put the chips back in the right places. Remove them (all) with your flatblade anything; pry them up gently, a little bit at a time from either end, alternating, or you'll bend the pins on the opposite end. Note that bent pins aren't the end of the world; some needle-nosed pliers will fix them handily. However, you can only bend them and fix them so many times before they break off, which *is* the end of the world.

Put the ROMs into the conductive foam that came with the SST (also keep the extra 68000 and oscillators in it too). Or, put them into some aluminum (cooking) foil and wrap them up; this is also pretty good protection against static. Be sure to keep the sketch with the TOS ROMs, so you have it if you need it.

If ever you need to re-install your 68000, you will *also* need to re-install these ROMs. The sketch you drew will help you get the TOS chips back in right.

Prepping the SST: Installing Those Expensive Options

Remove the SST from its static protection. (Remembering to practice safe static precautions!) Look for obvious things wrong - bullet holes through it, board cracked in two, another manufacturer's logo on it, static RAM caches. Nothing wrong? Ok, good. If there's something that wrong, just send it back to us, and we'll have some words with the carrier. We flog them when they foul up; keeps them in line...



Put the SST down on the table and have a look-see. First, check underneath to see that the connector pins are straight. Remember, you can straighten them if they are a little bent, or you can remove the protective socket (carefully) and put on another. Make sure that you put the conductive foam back on the pins (in order to protect from static and help keep the pins straight) while you mess with the parts on the top.

First, gently push down all the chips into their sockets, just in case they wiggled loose during shipping. Make sure you aren't squashing the pins underneath while you're doing it! If your SST has SIMMS installed, *don't push down on them*; just try to wiggle them and make sure they're tight. If one is loose, usually just gently pushing on it until the hold-down snaps in will do the job.

If you ordered your SST with Options direct from us, the Option parts will already be installed (and your SST thoroughly tested) when you get it. If you ordered from an Atari Dealer, they probably installed the Options for you. In either case, you can skip ahead to the next section. If you got the Option parts on your own, you will have to install them yourself. So, read on!

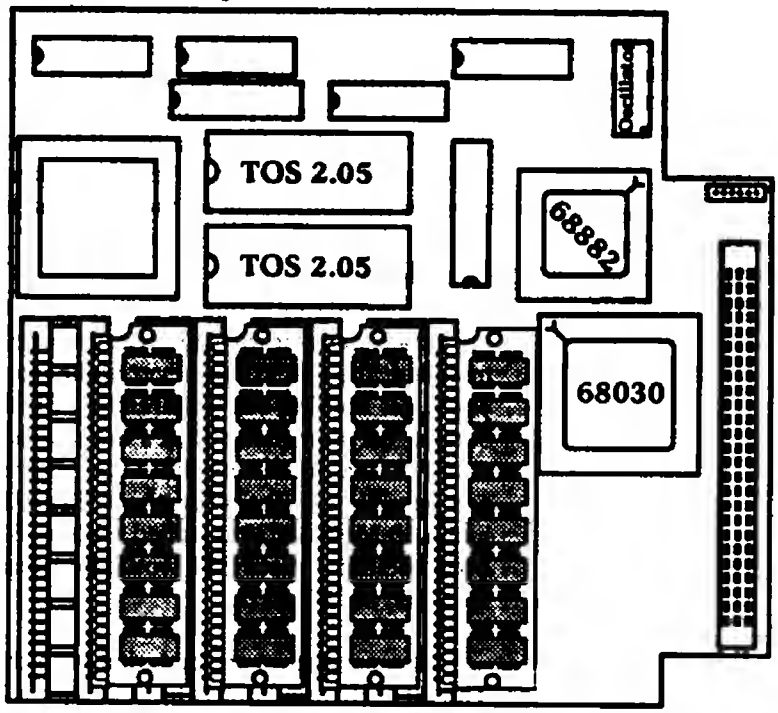
Installing the SIMMs

The SST board can be installed with 0, 4, or 8 megs of memory. It has to have those values because SST RAM is 32 bits wide, and each SIMM you put in is 8 bits wide. Four SIMMS thus make 32 bits of width. IBM style 9-bit SIMMS are perfectly useable; we just

ignore the 9th bit. ***It is much easier on your power supply if you use 1x8 SIMMs.***

You'll find out that it's pretty easy to remove or install a SIMM. To remove, gently pull apart the hold-downs on each side of the SIMM, one per side, and lift the SIMM up and out. To install, get the pins down into the socket, and gently press the chip down and in; you'll hear a horrible crackling sound, then the hold-downs will snap down. The SIMMs only fit in the socket one way. (Wiggle it a little to check if it's in solid.) This is all much harder to explain than it is to do; it's easy.

If you are installing only 4 meg of SIMMs, you have to put them in *specific sockets*; otherwise the SST won't work right. Start at the inside of the board, near the 68030, and put the first SIMM in the socket next to the 68030. Then put the rest of the SIMMs in, skipping every other socket; you should have an empty socket on the outer edge of the board when you are done. (Illumination 17.)



Illumination 17 - The SST with 4 meg of SIMMs in Alternating Sockets

If you upgrade from 4 to 8 megs, you'll need to pull 3 of the 4 SIMMs first, leaving the SIMM in the socket closest to the 68030 (insidemost socket). Then put the rest of the SIMMs in. It's *impossible* to squeeze a SIMM in between two others without damaging the SST because of the angling involved.

Basically, the *only* restriction on what kind of SIMMs you use are that they *must be page mode*. Also, be aware that the speed of your SIMMs (measured in nanoseconds) will *directly affect* the "ultimate speed" of your SST.

The 68030 & 68882 Pressure Intensive Installation

A 68030 is only about an inch and a quarter square (3.5 cm), and has what seems like a forest of a zillion gold plated pins. Actually, there are only 128 pins, but *each one* is just waiting to stop your heart by crumpling up when you push it into the socket.

When you get your 68030, it should come in conductive foam, and you should *leave it right where it is* until you take the static precautions we've mentioned before. If you don't, and it gets static zapped, you may be about to install a non-working 68030 into your SST.

Anyway, now that I've scared the "willies" out of you, let's start. On the top of the 68030 is a gold square with the part number and so forth printed on it. On one corner is a gold "Y" attached to the square. It kind of looks like a fish, with the "Y" part as the tail. That "tail" is actually showing you where to find pin 1 (there is also a tiny triangle which points to the exact pin). The 68030 needs to be installed so the "tail" is pointing towards the TOS EPROMs. (Illuminations 17 & 18, next page.)

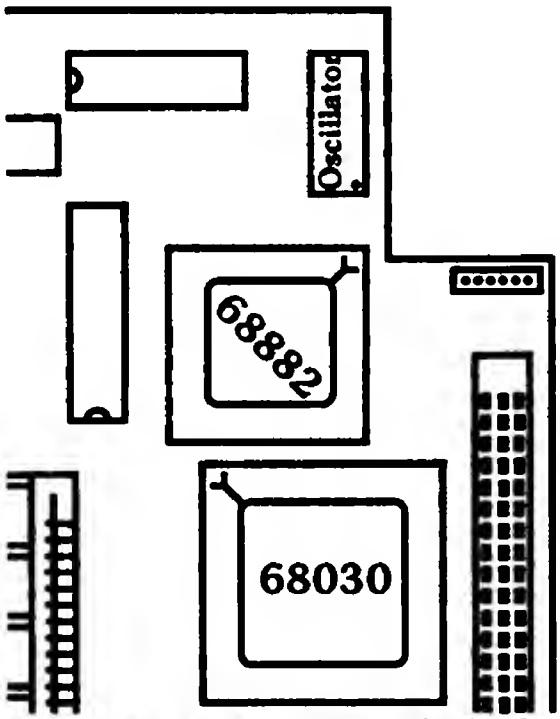
Position the 68030 *properly* (tail towards TOS), and *make sure* you have the pins lined up with the holes in the socket. Then push. Hard. This will start the pins, and give you a chance to see if any pins are bending. In order to push the 68030 *all* the way in, you will probably need to set the SST on the edge of a table with the connector pins underneath hanging out into empty space (so they don't get smashed flat). You should also cushion the table with a magazine or something, so it doesn't get scratched by the SST.

Then all you do is push on the 68030. Try to alternate the sides you push on, so the 68030 doesn't go in crooked. Push real hard. It *takes a lot of pressure* (80 pounds?) to get all

128 pins in far enough. You are done pushing when you can see only about 1 mm (or less) of the pins between the socket and the 68030.

Blood sacrifices are not required, but the soldered side of the SST board can be sharp, and you can really prick your fingers when you're starting the 68030 or 68882. I can speak from experience, like Sleeping Beauty.

Installation of the 68882 (or 68881) is basically the same, except the "tail" needs to be pointing towards the Oscillator, not the TOS. (Illuminations 17 & 18.) Also, it doesn't take quite as much pushing, since there are only 68 pins. As when you install the 68030, ***make sure you do not bend the connector pins on the underside of the SST board!***



Illumination 18 - 68030 & 68882 "Fish Tails"

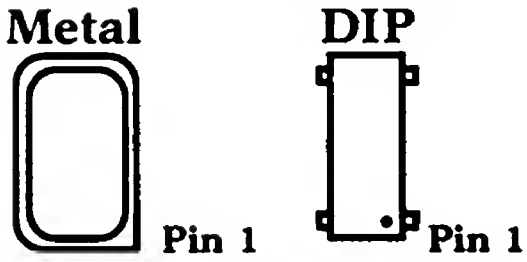
Oscillators

The oscillator is one of the most important parts of the SST; without one installed, the 68030 can never wake up, and you'll get nothing but the dreaded "Black Screen" on your monitor. It's the *controlling device* which determines how fast your SST can possibly go; like a heartbeat. Even if you had a 50 MHz 68030 installed with a 33 MHz oscillator, your SST would be running at 33 MHz, *not* 50 MHz.

Two oscillators are included with the SST: a 20 MHz and a 33.3333 MHz oscillator. The 20 MHz oscillator is for use with a 16 MHz 68030. The 33.3333 MHz oscillator is used with the 33 MHz 68030. (Very straightforward, right?)

Now for the tricky part. Let's say you have a 25 MHz 68030; which oscillator should you use? You can use the 20 MHz, but if you want *maximum speed* from your SST, your best bet would be to *get another oscillator* with a value somewhere between 23 and 28 MHz.

Aha, you say! Can I use a 33 MHz rated 68030 with a faster than 33.3333 MHz oscillator? Yup. We run SSTs around here with oscillator values up to 40 MHz, but since not all ST's are created equal, a safe value oscillator for you to get would be somewhere around 38 MHz.



Illumination 19 - Oscillator Packages

The kind of oscillator that works with the SST comes in two package types. One package looks a lot like a standard black plastic 14 pin DIP (Dual Inline Package) chip, except with 4 pins instead of 14, and the other is silver tone metal, again with only 4 pins. As with the 68000 (and most other chips), pin 1 on the DIP package oscillator is next to the dot. Pin 1 on the metal oscillator is on the square corner; the other corners are rounded. (Illumination 19.)

As before, take static precautions, and then install the oscillator you decided to use. Pin 1 should be on the end closest to the 68030. (Illuminations 18 & 19.)

Putting it all Together

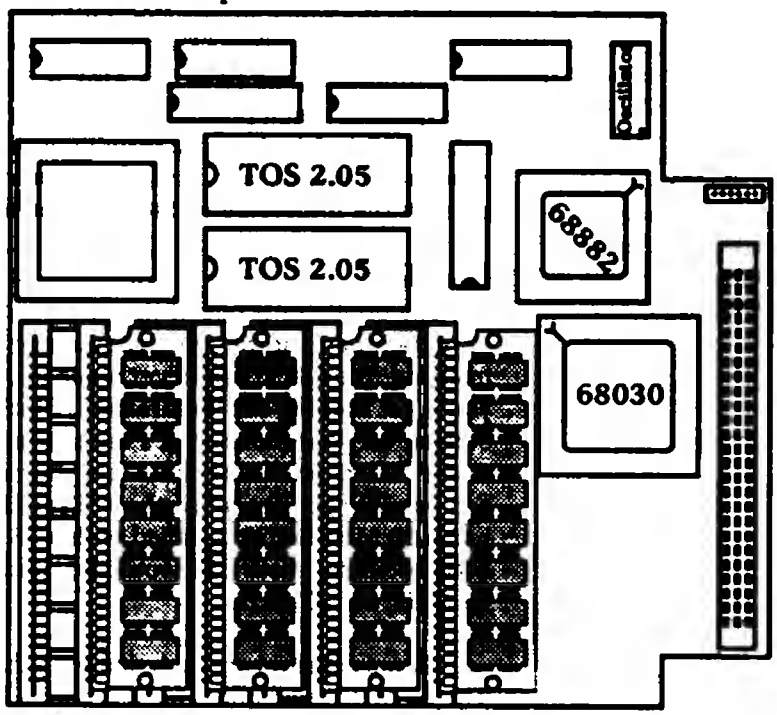
Okay, you have the SIMMs, the 68030 (and 68882), and the oscillator in the way you want them. Your ST should be off at this point, since you presumably don't want to light yourself up with it. Now we need to equalize the static between the SST and the ST and you. So, touch the SST and the ST at the same time (on that ground place I told you about).

ignore the 9th bit. ***It is much easier on your power supply if you use 1x8 SIMMs.***

You'll find out that it's pretty easy to remove or install a SIMM. To remove, gently pull apart the hold-downs on each side of the SIMM, one per side, and lift the SIMM up and out. To install, get the pins down into the socket, and gently press the chip down and in; you'll hear a horrible crackling sound, then the hold-downs will snap down. The SIMMs only fit in the socket one way. (Wiggle it a little to check if it's in solid.) This is all much harder to explain than it is to do; it's easy.

If you are installing only 4 meg of SIMMs, you have to put them in *specific sockets*; otherwise the SST won't work right. Start at the inside of the board, near the 68030, and put the first SIMM in the socket next to the 68030. Then put the rest of the SIMMs in, skipping every other socket; you should have an empty socket on the outer edge of the board when you are done. (Illumination 17.)

If you upgrade from 4 to 8 megs, you'll need to pull 3 of the 4 SIMMs first, leaving the SIMM in the socket closest to the 68030 (insidemost socket). Then put the rest of the SIMMs in. It's *impossible* to squeeze a SIMM in between two others without damaging the SST because of the angling involved.



Illumination 17 - The SST with 4 meg of SIMMs in Alternating Sockets

Basically, the *only* restriction on what kind of SIMMs you use are that they *must be page mode*. Also, be aware that the speed of your SIMMs (measured in nanoseconds) will *directly affect* the "ultimate speed" of your SST.

The 68030 & 68882 Pressure Intensive Installation

A 68030 is only about an inch and a quarter square (3.5 cm), and has what seems like a forest of a zillion gold plated pins. Actually, there are only 128 pins, but *each one* is just waiting to stop your heart by crumpling up when you push it into the socket.

When you get your 68030, it should come in conductive foam, and you should *leave it right where it is* until you take the static precautions we've mentioned before. If you don't, and it gets static zapped, you may be about to install a non-working 68030 into your SST.

Anyway, now that I've scared the "willies" out of you, let's start. On the top of the 68030 is a gold square with the part number and so forth printed on it. On one corner is a gold "Y" attached to the square. It kind of looks like a fish, with the "Y" part as the tail. That "tail" is actually showing you where to find pin 1 (there is also a tiny triangle which points to the exact pin). The 68030 needs to be installed so the "tail" is pointing towards the TOS EPROMs. (Illuminations 17 & 18, next page.)

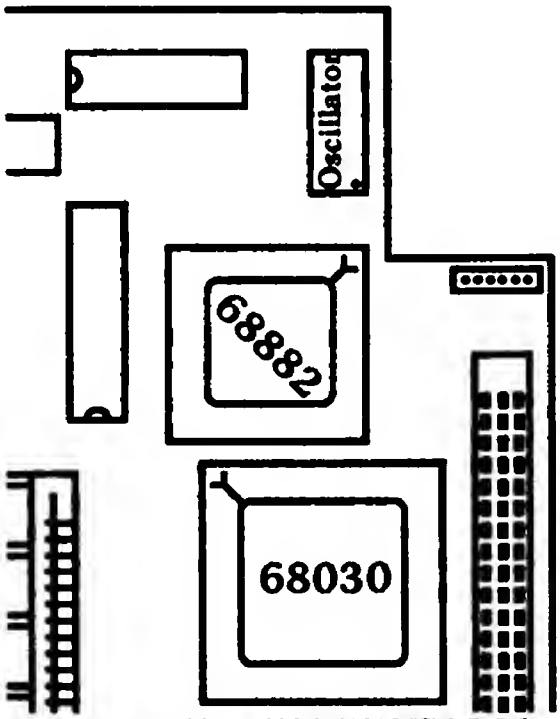
Position the 68030 *properly* (tail towards TOS), and *make sure* you have the pins lined up with the holes in the socket. Then push. Hard. This will start the pins, and give you a chance to see if any pins are bending. In order to push the 68030 *all* the way in, you will probably need to set the SST on the edge of a table with the connector pins underneath hanging out into empty space (so they don't get smashed flat). You should also cushion the table with a magazine or something, so it doesn't get scratched by the SST.

Then all you do is push on the 68030. Try to alternate the sides you push on, so the 68030 doesn't go in crooked. Push real hard. It *takes a lot of pressure* (80 pounds?) to get all

128 pins in far enough. You are done pushing when you can see only about 1 mm (or less) of the pins between the socket and the 68030.

Blood sacrifices are not required, but the soldered side of the SST board can be sharp, and you can really prick your fingers when you're starting the 68030 or 68882. I can speak from experience, like Sleeping Beauty.

Installation of the 68882 (or 68881) is basically the same, except the "tail" needs to be pointing towards the Oscillator, not the TOS. (Illuminations 17 & 18.) Also, it doesn't take quite as much pushing, since there are only 68 pins. As when you install the 68030, ***make sure you do not bend the connector pins on the underside of the SST board!***



Illumination 18 - 68030 & 68882 "Fish Tails"

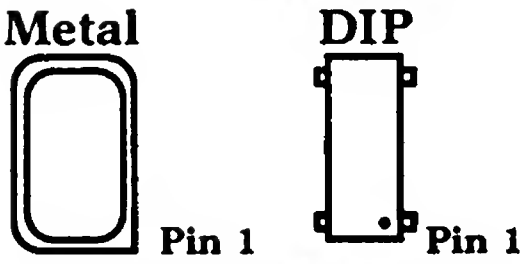
Oscillators

The oscillator is one of the most important parts of the SST; without one installed, the 68030 can never wake up, and you'll get nothing but the dreaded "Black Screen" on your monitor. It's the *controlling device* which determines how fast your SST can possibly go; like a heartbeat. Even if you had a 50 MHz 68030 installed with a 33 MHz oscillator, your SST would be running at 33 MHz, *not* 50 MHz.

Two oscillators are included with the SST: a 20 MHz and a 33.3333 MHz oscillator. The 20 MHz oscillator is for use with a 16 MHz 68030. The 33.3333 MHz oscillator is used with the 33 MHz 68030. (Very straightforward, right?)

Now for the tricky part. Let's say you have a 25 MHz 68030; which oscillator should you use? You can use the 20 MHz, but if you want *maximum speed* from your SST, your best bet would be to *get another oscillator* with a value somewhere between 23 and 28 MHz.

Aha, you say! Can I use a 33 MHz rated 68030 with a faster than 33.3333 MHz oscillator? Yup. We run SSTs around here with oscillator values up to 40 MHz, but since not all ST's are created equal, a safe value oscillator for you to get would be somewhere around 38 MHz.



Illumination 19 - Oscillator Packages

The kind of oscillator that works with the SST comes in two package types. One package looks a lot like a standard black plastic 14 pin DIP (Dual Inline Package) chip, except with 4 pins instead of 14, and the other is silver tone metal, again with only 4 pins. As with the 68000 (and most other chips), pin 1 on the DIP package oscillator is next to the dot. Pin 1 on the metal oscillator is on the square corner; the other corners are rounded. (Illumination 19.)

As before, take static precautions, and then install the oscillator you decided to use. Pin 1 should be on the end closest to the 68030. (Illuminations 18 & 19.)

Putting it all Together

Okay, you have the SIMMs, the 68030 (and 68882), and the oscillator in the way you want them. Your ST should be off at this point, since you presumably don't want to light yourself up with it. Now we need to equalize the static between the SST and the ST and you. So, touch the SST and the ST at the same time (on that ground place I told you about).

Next, see the connector pins on the bottom of the SST? That connector needs to go into your ST's 64-pin socket. Now, your SST should have an extra socket on to protect the SST's non removable pins (I always try to keep at least one removable socket on the SST; sockets are easy to swap if a pin bends or breaks, and that happens to everyone on a bad day.).

Editor: Especially if that day happens to be a Monday!

Towards the front of the ST is the MCU chip, often with a big metal clip on it. You *don't want* this metal clip to touch the underside of the SST, and *it will unless you do something about it.* You can't remove the clip, which corrects for socket weaknesses; however, you can slip an insulator under the board. It has to be thin or it will force up one side of the SST's connecting socket.

Some SSTs may have a cardboard insulator already installed.

I've found that thin cardboard, a 5 1/4" floppy disk jacket, and many other ***non-conductive*** things make fine insulators. Tape will do if you're *really* without anything else. However, while the MCU is the highest chip in the region, we don't want the SST touching down anywhere else, either; so it's really best to use a larger insulator.

Put the insulator in place (taping it down a little is very helpful, so it doesn't move), and plug in the SST. Once you have the pins lined up, you will need to press very, *very hard* to get the SST "properly seated in". ***Make sure all your pins are lined up first!***

If you'll get your head down at the ST socket level, you can guide the SST pins in. Looking from the back or the front or the side all seem to work when I do it. Take your time and check that you don't have it in "one pin off", with one pin dangling in the air. Start a *corner* of the row farthest away from you; then line up the back row. With the board tilted up gently start the pins you lined up, then rock the board forward and start the other row. It's really pretty straightforward. A flashlight can also be a big help here for a final check that all is well.

It won't exactly "snap" down into the socket, but it *will let you know* when it's firmly in; you can push down on the board pretty hard above the socket (next to the two SST TOS ROMs). ***Please don't push on the SIMMs; those sockets will break.***

Well, that's it. It's ready to test. I would not recommend putting the top shield and cover back on until you've checked it out; if there's *any* problems, you want to be able to get at the SST board.

About Your Power Supply

We've run 8 megabytes worth of SIMMs on a Mega ST for 6 days with no overheating or burnouts. SIMMs are Dynamic RAM, and DRAM is fundamentally a bunch of extraordinarily leaky memory cells that have to be continually refilled (refreshed); the fill is part of the power draw, as are the output "drivers" that push data onto the data bus.

On the SST with 8 megabytes of SIMMs and running a burst mode test loop (where the DRAM is working flat out, pulling the maximum amount of current), the entire machine (ST & SST together) pulled a maximum of 2.3 amps. Mega ST power supplies are rated at 3.0 amps, since it was designed to be expanded. So, if all you have installed in your ST is the SST, you have no need to spend nights worrying about it.

However, the Moniterm video card is a power hog. It pulls a lot of amps, just by itself! You should add an external power supply or install a higher rated power supply inside your Mega.

A common symptom of things going sour in the power supply department is trying to load a program into SST RAM and having the machine lockup or crash, when you darn well *know* the program is okay to run in SST RAM.

Please realize that some programs don't like running in SST RAM *at all* (most seem to like it okay), and will crash *every time*. A power supply problem is a possiblility if a program will run one day, then fail the next day.

Based on our experience, you'll have no power supply problem at all with 4 or 8 megabytes. Should things start getting a little flaky at 8 megs, go back to 4 megs, or you could add a Power Supply, which supplies the SST's entire power needs and lets your ST's supply handle only the ST. (In other words, call us, or George Richardson.)

Caution: Incompatible hard disk drivers or not running the DISKxx program when it is needed also can result in this type of problem.

Okay, installation is basically complete; let's continue on, and check your SST out.

It's Show Time!: A Tragicomedy

Computer Trade Shows

They're real, real different when you attend one, versus when you **work** a booth at one. **Real** different.

It's definitely a better bet to **attend** one. There, you get to see everything spiffed up, pretty, "ready to go", literature ready, everyone in nice suits and ties or in "power business" outfits (well... except that Dave Small character in his hacker T-shirt), and see the carefully planned, working demonstrations, designed to make you feel that **this** product is something you simply must buy before you die of old age.

And, funny of funnies, there are usually a stack of shrink-wrapped Products sitting in the booth, giving the real impression that the product is "done". Ha!

Having worked booths at computer trade shows, I'll tell you one thing they all have in common. The people doing the demonstrations have all been **carefully** briefed on **what not to demonstrate**; e.g., what crashes the machine.

Me? Did I do this? Well, instead of avoiding crashes, I had fun with them instead. With the Macintosh Emulator, I tuned it so that after a crash, the slightest wobble of the mouse would "push the RESET button" and reboot the machine.

Then, during the demonstration, I'd go into a program that crashed the emulator. The depressing "crash screen" would plot. I'd then say, "Any program crashes at some point. But we alone build in a frustration reducer. Watch!".

And I'd hit the table with a balled-up fist.

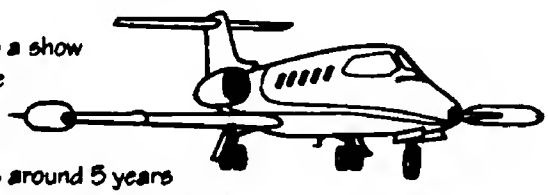
In turn, that would joggle the table, which joggled the mouse, which RESET the machine and brought it back up. To the person I was giving the demo to, it darned well looked like hitting **the** table caused the restart. (Very few people know just how sensitive a mouse is!)

It was a never-fail, absolute killer demo, the best I have ever done)

Now it's definitely unbelievably exhausting to **"work a show"**. This is something we at Gadgets don't lightly undertake anymore – and believe me, the cost (averages a thousand dollars or more per show) is the least of the worries.

First off, the computers and equipment for the show have to come from **somewhere**. I assure you we don't keep a set of equipment all boxed up, say, three working machines, hard disks, monitors, all with canned, bug-free (ha!) software and self-running demonstrations.

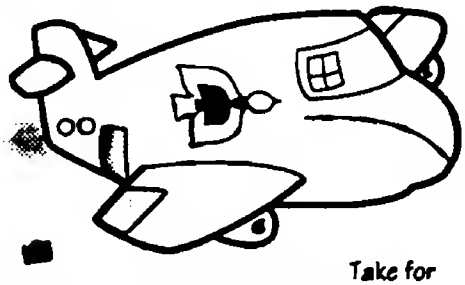
No; we yank the **working** machines right off our desks, because they are the only machines we've got, and take them. (I'm sure you're beginning to see the problems already.) But we just can't afford to have that much idle hardware laying around; it costs too much. (Atari Corp., by the way, usually supplies most of the Atari Equipment at shows – but they don't like us to open their machines and install a socket where their 68000 used to be!)



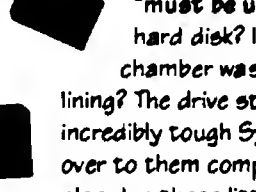
Taking a computer to a show invariably means an airline trip; the last Atari show within reasonable driving distance from Denver was around 5 years ago! So, we carefully pack everything up and hop into our corporate Lear Jet and... oops, we don't **have** a Lear Jet. So, we travel by commercial air.

It's Show Time!: A Tragicomedy

This, in turn, is a dreadful thing to do to a perfectly nice, friendly computer. Typically, the results are that after the computer arrives at baggage claim, it looks like it's been kicked out of the plane at 30,000 feet before landing, probably to speed up baggage service. And believe me – no amount of padding helps. In some impossible way, it almost makes things worse.



Take for instance those demonic "packing peanuts": the little pellets that people pack equipment in. During the flight they work their way into **everything**. Ever had a packing peanut inside of a **"must be ultra-clean dust free"** Syquest hard disk? I have – in fact, the whole chamber was crammed full. (The silver lining? The drive still worked, proving how incredibly tough Syquests are. I have switched over to them completely and am glad I did.) I've also dug these little peanut devils out of Mega ST's, Stacy machines (how?!?), MegaFile hard disks, and so forth.



As the result of going to these shows, I now have one crippled Mega ST. Mind you, it can be fixed if I ship it (see: Airline Baggage above) to a hot ST repairman... it works fine after George has put, oh, ten hours and seven new chips into it. Problem is, it simply can't survive the trip back to me from George! After umpteen million shows, and three round trips for repair, I have retired it with thanks; it now occupies a place of honor in our basement, the only horizontal surface unused, yes: the top of the kitty litter box.

The other two Mega machines (I'm not even telling you how many 520 and 1040 or TT corpses are on the premises!) are in a constant "Going Up! Going Down! Going Up!" status. I honestly wonder, every day as I enter the office, if the machines will start up today.

Editor: It's just Monday-itis. They all work fine on Tuesdays!

Other Editor: That's because I spend all day Monday fixing them!

Kids and Shows

Since these computer shows happen all over the world, we usually take our kids, Eric, Jenny, and Jamie (who's been doing computer shows since the age of 5 months) along. That way, they get to see places they haven't been to before. They've probably been on more airplanes than most kids their age!

Let's see... 3 times to Disneyland and Universal Studios, twice to Washington DC to see the white marble monuments and calcified Senators... and the National Air and Space Museum at the Smithsonian. Detroit (we drove into Canada). Once to London (a thousand year old Castle), twice to Germany (they got to see the crumbled remains of the Berlin Wall).

Yes, it's harder bringing the kids to computer shows. No more "conferences" at the local bar until 2 AM, with a hangover the next day. No late-night meetings in smoke filled rooms. It's tough. *sigh*

But it is **awfully nice**, when driving along the Autobahn (or some other road), to be able to point and say "WOW. Look at THAT!" And see the kids; their eyes glowing with excitement... instead of just showing them the pictures weeks later.

Back to the Show

Arriving at the show with airline-shipped computers is like a gigantic dice roll... with rigged dice. Now usually, you set up your company booth (see above description) the night before the show. This is where you pray, unpack everything, pray, put it together, pray, plug it all in, pray, and test it out. Invariably, several critical things fail, and you have to fix them, with only your wits (very frazzled by the plane trip – what is it that's so tiring about sitting in a plane? Oh, but it is... It is...) and whatever tools you brought.

Editor: It's bringing the kids, and 12 million trips to the bathroom (to watch the toilet flush).

Kludging is the **rule**, not the exception.

As we've gone to more and more shows, the toolkit has gotten more extensive. I would describe it as "out of control" now. We now have an entire batch of floppies we hand-carry onto the plane (braving the metal detector magnetic fields, which are no joke), called the Show

Survival Kit. I tell you, if the machine can even wake up for a few instants, if that hard disk can even rotate a bit, those disks can usually cure what's wrong. They have amazing, near sorcerous abilities in them. Many a sinful hard disk setup has felt the wrath of the Show Survival Kit and mended its ways.

Ever arrived at a show and found that there were going to be "loaner hard disks" available... with an interface that every single one of your demonstration disks is incompatible with? I have. I get out the Show Survival Kit. Ever had a demo disk get damaged? I have. Copy off all you can with the Show Survival Kit (Partcopy in particular.) Ever wonder which disk cable is flaking out, gone bad? Dig out SUPEDIT in the Show Survival Kit, start accessing the hard disk, and start wiggling cables; when you hit the bad one, it'll show.

Sometimes, not all the tools you can bring will get you through, and you're left with only your wits. Now as someone who is 33, I will cheerfully tell you that wits are in short supply when I'm working a show alone... it's my age, you see. (Close to retirement for a Hacker!) For instance, at a show not too long ago, I brought two computers. Both were Dead-On-Arrival (of course, they worked before I airline-shipped them.) I recall getting one machine working through sheer cannibalism of the other machine and flexing its circuit board to encourage cracked traces to come together. But the other machine? I proudly left it out on the tabletop, all opened up, and said, "This is a computer I brought just to display how our SST board fits inside. Of course, I haven't plugged it in; that wouldn't be safe, with the power supply out in the open and kids here at this show."

No one blinked an eye. "Oh. Ok."

You do the best you can, folks; and the other exhibitors, fighting crashes, power line spikes and dropouts, crashed disks, and people doing the weirdest things to their computers, are in the same boat. (In fact, it tends to bring the exhibitors together emotionally. Equipment and labor is freely lent out and swapped around as needed. During one show my entire aisle, except for the Gadgets booth, lost AC line power; we held a fast collection of power-strips, plugged everyone into the Gadgets booth, and wonder of wonders, got everyone back working inside of 15 minutes.)

Editor: The "modus operandi" of equipment at computer shows seems to be that no matter how much time and effort you put into getting it up and running before the show, it only starts working 10 minutes after the show opens. For no particular reason.

Well, whether or not you have things fixed (and many times you don't), the show opens. Invariably 78 people begin to talk to you, each interrupting the other. Heck, if you were a computer, you'd stack-overflow and crash from the interrupt conflicts; since you're a human, all you can do is wish you could hide somewhere, pass out literature as best you can, and try (and fail) to have a civilized conversation with one person at a time.

I understand that people want to see the whole show and ask questions... but why are all manners suspended? Why does it become a good thing to be the person who interrupts other people the most? I feel bad for the poor souls who patiently wait their turn, for sometimes they don't get a turn. (I try very hard to pick them for "Next question?".)

The ordeal lasts about a year, and then it's over until the next day. Did you remember to get a hot dog (or some other culinary delight) for lunch? You have to hide to eat it! Otherwise, people will be shooting questions at you while you're eating, and not minding at all that you're answering with your mouth full, spraying them with ketchup, mustard, bread, and whatever they make hot dogs with. It doesn't seem to bother the questioner a bit. (On the other hand, anyone tacky enough to interrupt someone's lunch probably deserves what they get.)

The absolute low point (of any show I've ever

It's Show Time!: A Tragicomedy

been to) happened to me in Washington DC, when I went to the bathroom to take a very necessary break. Three very determined people followed me right in, shooting questions at me about Macintosh emulators, literally as I was standing there, errrr, doing my thing. It really happened, and yes, nowadays I hide in a stall.

So you endure. After all this fun, a ragged cheer erupts from the exhibitors when the first day of the show is officially over. You may think we then go out carousing. It's true; I carouse right over to the nearest food place, then carouse straight to bed. I am flat exhausted.

The second day, you not only have Show Fun to look forward to, but you have Packing Up to do, just when you're completely exhausted at day's end. This is where you use yards (meters) of packing tape to try to make the boxes which you brought hang together through the airline's worst treatment for the trip back. Sometimes this works. Sometimes. Then, you try to remember all the details of getting out of the show (did you borrow any equipment? Who has your spare power strip?) At the door, Hey, You! Prove this is your computer, not one from The Corporation! That's right, unpack it (thus slicing through all your yards of tape).

Then, repack it again ("sigh"), lug it to the rental car, get back to the hotel, pack up there, check out, get to the airport (the hardest time I ever had at this was in Burbank, a relatively small airport; I went three quarters of the way around it before I could find a way in), check in the rental car, somehow get all the equipment to the airline counter and checked-in... and wonder what you forgot this time.

Now, you might think this is just a digitized picture enlarged too much. In actuality, we're just at a Computer Show, and feeling a little "stretched".

Now these are two day shows, which are the most common these days in the Atari community. But there are also 5 day (Comdex) and 8 day (CeBIT) shows. Those are killers.

I have never approached them with anything but horrified dread, and I have never been disappointed. A two day show you can do more or less off of reserve energy; you don't really need that hot dog for food. Not so when the show is a week long.

First off, anyone of importance attends only the first few days, and that's it. The shows are deserted the last days, exhibitors standing around like shy kids at a dance. The last day, people often begin packing up around noon.

Second, with shows that big, there are so many people that the Interruption Factor becomes insane; it's almost impossible to give a simple demonstration of anything.

Third, the piles of literature you have to take to such a show are incredibly heavy. And there are regulations about using little wheeled carts to help you move these 80 pound (30 kilo) boxes around if you are not part of the show's staff, so you have to carry them in by hand. Believe me, it overwhelms any anti-perspirant in the world by the time you've carried in the last box.

Fourth, the piles of literature that showgoers gather at a big show could break the proverbial camel's back. When I go to shows, I point-blank refuse any hand-out literature or buttons, I give them a business card, and ask them to mail me literature. If literature ever shows up, I know they're somewhat on the ball... maybe this is a company I should deal with! If not, oh, well. (And yes, when I'm working a booth, I end up with about a foot-high stack of business cards, with "mail literature to" scribbled on them.)

(One intelligent company threw a party and invited the press, which is ho-hum, but had a new angle: they offered to express-ship your gathered literature back to your magazine office. Their party was an overwhelming success!)

After days of working such a show, you go numb inside. It's less than fun, and honestly, an idiotic way to do business. The ancient Chinese curse says, "May you live in interesting times"; Computer Shows are certainly Interesting Times.

Pre-Flight Checkout

During pre-flight checkout, be sure to remove the two ejection seat "ARM" Safety Clips, located at the base of the chair on each side of the cockpit. Also ensure that you have been refuelled with JP-6 fuel; standard JP-7 does not work well.

Power-up Video Check

Okay, you've got the SST board plugged into your ST. Let's check it out. This comes in two steps:

- See if the ST and SST power up normally, and
- See if the SST's fastRam works right.

Make sure you have reattached the power supply, monitor, and floppy disk drive properly. (This reminder is for those of you who decided to test their luck and skipped the test with the 68000.) It's awfully embarassing to get the dreaded "black screen" (no video) simply because the monitor isn't plugged in!

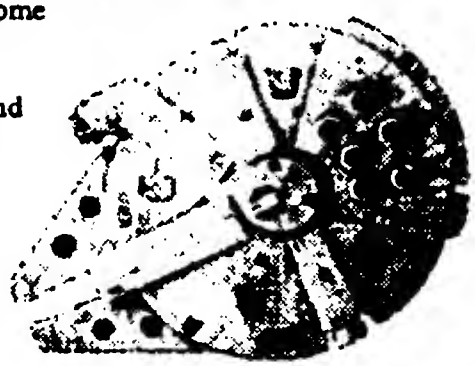
Editor: Dave speaks from experience. He did it while testing MegaTalk boards, and it took him awhile to figure out why so many of them were doing the "black screen"!

Other Editor: Hey, Editor! None of that!

So, hook up any cables that need to be plugged in (video, keyboard, power...) and let's go for it!

Go ahead and turn on power. ***You'll want to be watching your monitor.*** When the ST with SST is first powered on, it runs the program in the TOS 2.06 ROMs, which does a few RAM checks, then turns on the video output. For instance, on the monochrome monitor, you get a bright white screen. That's a sign that the SST is at least *trying* to work, and that the CPU is running the program in the TOS 2.06 ROM onboard the SST. If you get no video in 3-5 seconds, shut off the ST and recheck *everything*.

"Chewie, it's not my fault!"



If you get normal video, you can just skip over the the whole next "troubleshooting" section.

Troubleshooting: The Warp Drive Isn't Operational

Ok, take a deep breath, and *do not panic*. Most problems are really simple and easy to fix. We've found that despite their complexity, SSTs can stand amazing abuse, and end up working fine. Let's check a few things. Remember to take static precautions, and make sure your ST is *OFF*.

If your video turns on but is all scrambled up, and you have a BLITTER on, it's an easy fix. Remember that floppy disk you (were supposed to have) made before you took apart your ST? The one that has the BLITTER turned off in the Desktop Information file? Use that floppy to start up with the BLITTER off. Some BLITTER's just can't handle the SST unless they are turned off. With a BLITTER on, results vary considerably, depending on what program you are running. We recommend not worrying about it, and just leaving the BLITTER off.

If you have a "black screen" the first thing to try is power off and unplugging the hard disk cable (the DMA cable) from the back of your ST, and *then* try turning on your ST. (Some hard disks conflict with the RESET line, so the 68030 never gets it's "wake up call", or worse, gets it at the wrong time.) If you get no video in 3-5 seconds, shut off the ST; it's not the hard disk RESET problem. If you get normal video with your hard disks unplugged, you will need to do the RESET line fix.

There's two ways to do this fix: the quick and dirty (and not always reliable) way, and the right way, which involves cutting traces on your ST motherboard. For the quick and dirty way, just solder a 1K resistor between pin 2 of the 7407 and pin 14 of the 7407. (Illumination 20.) The 7407 is labeled U2 on the ST motherboard, and is located *under* the power supply. ***This will not solve everyone's hard disk problem;*** for that you *have* to do George Richardson's complete Hard Disk RESET Line Fix, detailed in a later chapter.

Illumination 20 - Quick & Dirty Fix

Checking the SST Board

Try reseating (removing and putting back in) the SST board; it may have been a "pin off", or there might have been some debris in the socket, or it may not have been all the way in. Also, check to see if any of the connector pins are bent. Reinstall the SST, and try it again. If you still don't have video, turn the ST off, and we'll try reducing the load on your power supply.

If you have 8 megabytes of memory on the SST, we *highly recommend* that you use 1x8 SIMMs, not 1x9s. It can make a *big* difference to your power supply. Try removing all of the SIMMs, or just the high 4 megabytes. (Remember, the SIMMs are high-low-high-low order; if you remove the high 4 megs, you'll have none-low-none-low ordering, from left to right, as shown in Illumination 17 earlier.) Removing 4 megs will help ease the load on your power supply.

If any of the above fixed the video, continue on to "Warp Drive is Operative".

Editor: You passed "GO"; collect $200 (from the game Monopoly™).

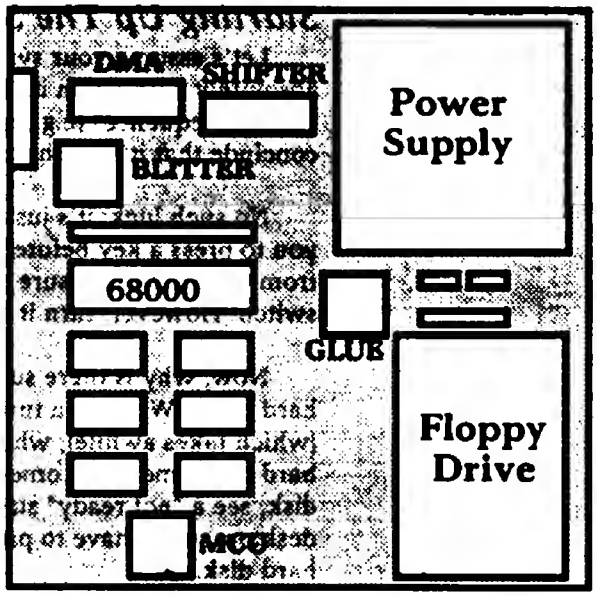
Check the ST Without the SST

If it still won't come up, let's see if the *ST* itself is okay. Remove the SST, put in a 68000 and the old TOS ROMs, and switch it on. If it's still dead, you've got a "dead ST" which it's time to check out. This happens from time to time; sometimes just opening up the ST seems to kill it, weirdly enough (actually, there's reasons related to circuit board flexing that are the cause). Other times static electricity is the culprit, or oxidation (or our blasted cat PyeWackette, who walks across our equipment with her fur crackling with static). Now, it's *usually something easy to fix*, like reseating the MMU and GLUE, SHIFTER and DMA chips - anything on the ST that's socketed.

What we do here is try to scrape off the crud ("oxides") that develops on the pins of the ST's regular chips, and on the sockets, by picking the chip up some and sliding it back down. The crud-layer ("oxidation depth") is very thin, and this scraping will cut right through it. That re-establishes contact. This is one reason there are so few sockets on the ST; each one is a potential problem. Reseating chips is SOP (Standard Operating Procedure) for many, many ST owners! This is not an SST-only kind of thing. Illumination

21 shows the locations of the Mega ST chips you will need to reseat: DMA, SHIFTER, GLUE, and MCU.

Note: I have tried "Tweek!™" (and other "contact-enhancers") on a 520 with crud-itis. My machine then would not start. I had to remove the Tweek! with an ultra super duper warp drive cleaner, "Piranha!" (which will just about strip your fingers to white, polished bones if you touch it) before the 520 would restart. I suspect that the ST was so near to the edge that Tweek! just wasn't right for it. Tweek! works on other computers I have; I can't recommend it for the ST though!



Illumination 21 - The Mega ST Chip Locations

Reseating DMA and SHIFTER

These are long, rectangular chips like the 68000, but smaller. With the long, skinny chips that are socketed, a screwdriver will enable you to gently lift the chip up, a side at a time, and then you push it down hard to reseat it. I *never* completely remove the chip; this prevents many reinstallation problems like pins-up, getting the chip reversed, and so on. Also, if you pry up one side completely, you will have bent the pins on the other side; so be careful! A little at a time does it. Pry up one side a millimeter, then the other side, and repeat, until the chip is pretty loose (up about 1/8" or 3 mm), then *push* the chip into the socket.

Reseating PLCC's Chips: GLUE and MCU

With the 1" (2.2 cm) square "PLCC" chips (GLUE and MCU), you're in for a harder time. What you want is a PLCC "puller"; they're about five dollars. However, no one seems to have them when you need them! I have found a way, though. If you get a tool with a very sharp end (like an X-acto knife), you can sometimes use the notches in the PLCC socket's corners and "scrape against" the chip hard enough to lever it up; digging into the chip to get a grip on it. I do it with the X-acto blade held vertically, "digging in" to the PLCC chip with the point (if you do it horizontally, you'll break the X-acto tip, and possibly the socket).

With these PLCC's, do not try to pry it up directly, even with a tiny screwdriver, going in through the notch; you will break the socket, at which point, you're in for a costly repair bill. (I know about this one the hard way.)

Note that Atari often puts steel brackets (hold-downs) on top of the square chips to help hold the sockets together. (You will want to re-install them, or the socket will deform and lose contact.) Think about this: If Atari had to install reinforcers for those sockets, do you want to be prying against them? They crack *so* easily...

Okay, now re-test the machine with the original 68000 and TOS ROMs. Does it work? Doublecheck again to make sure all the cables are hooked up correctly, and are pushed in tightly.

If it still doesn't work, the ST is probably sick and needs the attention of a good serviceperson at your Atari Dealer. We've found that usually in this case static electricity fried one of the big VLSI chips (like GLUE), and the only cure is replacement. There are several good Atari service centers around; see a later chapter for a list.

Retest the ST with the SST

Okay, we're assuming at this point that your ST is working without the SST installed. If the ST comes up to video and generally seems to work okay, then it's time to re-install the SST and try it again. Don't be surprised if it works fine this time around; I can't tell you how many times I've had to reseat chips in my ST to get it to work!

If the machine still won't even kick on video with the SST attached, then it's time to call or fax us, and we'll try to figure out what's wrong. All the boards are tested before they're sent out, so there might be some interaction between your ST and the SST we can work out over the phone. Also, we can't put every possibility into this manual, or it would be way too long!

Starting Up The SST: Warp Drive is Operative

Let's assume your system came up and you're staring at this exciting lit-up screen with absolutely nothing on it. The system is going to sit there looking at you, not beginning the "bootup sequence" (e.g., not plotting a desktop or hitting a hard disk) and you're going to conclude that it's crashed.

No such luck; it's just being a TOS 2.05 machine. Both Mega ST* and the TT require you to press a key before they will begin booting up! You'll find yourself forgetting this from time to time; I sure do. I expect a machine to start up when I turn on the power switch! However, turn it on, pause, press a key, and your system should start up.

Now, why is there such a delay? It's for the Mega ST* and TT, which have internal hard disks. When you turn on those systems' power, the hard disk starts spinning up (which takes awhile), while the CPU is ready almost instantly. The delay is to give the hard disk time to become ready. If there wasn't a delay, the system would look to the hard disk, see a "not ready" status, think, "Hmmm, no hard disk", and plot a two-floppy-disk desktop; you'd have to press RESET *after* the hard disk was ready, in order to access the hard disk.

MAKEDSK.PRG

There is a program called "MAKEDSK.PRG" on the SST Release Disk which makes a floppy disk you can leave in Drive A:. It has an executable boot-sector that does all the right things to bypass the delay. The delay will shorten from 90 seconds to 2 seconds if you have this MAKEDSK floppy in Drive A:.

TOS 2.06 (which you will have on your SST if we're *really* lucky) has a completely different startup sequence. It plots an Atari logo and sits there doing a ROM test, then a memory test, then... You can abort these tests by pressing a key, or by using the MAKEDSK floppy.

Getting your Hard Disk Up and Running

If you have a hard disk hooked up, it should start up normally. However, sometimes it will not, because *now* we're running a fast 68030. Right now, if you see some signs that it's trying to boot (text onscreen, for instance), you're doing ok. With an ICD host adaptor, you need the 5.4.2 (latest as of this writing) software to use the SST properly, Atari's HDX 4.03 or 5.0, or Supra's 3.43; these are the *only ones we know of* that handle the speed of the 68030 and SST RAM loading correctly. Previous versions have a tendency to sputter, wheeze, and fail.

This means you need to get a copy of ICD's newer software or Atari's newer software that is 68030 (e.g., TT) compatible. ***In general***, when something is "fixed" for the TT, it's also fixed for SST. You don't *have* to use Atari or ICD's software, just hard disk software from your Atari Dealer that is "TT compatible"; we just know that these Atari and ICD versions definitely work, and also many "aftermarket" hard disks use ICD innards.

Editor: If they followed the Hard Disk and Installation Instructions, they should already have everything all set up and ready to go (but who reads computer manuals nowadays?)!

Refer to the earlier chapter "Hard Disks, Floppies, and the SST", and follow the instructions there to set your hard disk up properly for the driver software you are using.

If you're having hard disk problems, and want to proceed with the checkout anyway, turn your system off, and unplug the hard disk connector. (Easy enough.) We'll continue with the software on your backup of the SST Release Disk.

If you don't have a hard disk hooked up, the SST will look for something on the floppy drives, then shrug and give you the desktop. This is the new "TT-ST® Desktop", not the older Atari ST desktop; this new desktop can do all sorts of *new* things. It has all the editable icons and whatnot of the TT desktop. The only entry that really interests us now is the "CACHE"; we want to turn it on and make sure things happen faster. The "CACHE" on/off menu entry is where the "BLITTER" entry used to be, fourth menu from the left (OPTIONS), on the bottom.

Open up the floppy's directory window. Use the top, right-hand corner sizer button to make the directory full-screen, then back again. Now, go to the "CACHE" entry in the menus, select it (it will then show with a check mark next to it), and size the directory windows again. You should see noticeably snappier performance.

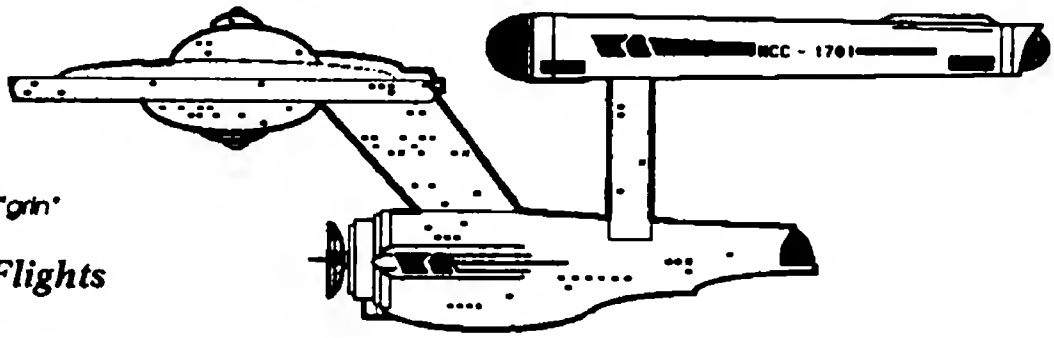
What we doing now is running in ST RAM, which ain't that hot, but using the cache of the 68030 and the higher speed of the 68030 to accelerate things. If you're running at 33 MHz, you'll find you're around 4 times the performance of an ST in many things. (Once you kick in SST RAM, that figures goes up to 8 to 12 times. That's the whole point of SST RAM!)

If you run without the CACHE on, you will run at essentially ST speed. If you run with CACHE on, you'll run at about 4 times ST speed, with some things slower, some faster, depending on what the program you are using does.

Okay, the machine is looking pretty good. The 68030 is running, TOS 2.05 (or 2.06) is up and running, and video and such is looking good. Now that the engine seems to work ok, it's time to test out the SST's "supercharger", SST fastRAM.

Editor: Supercharger? I think "Afterburners" or "TransWarp" sounds better! After all, it is the 25th Anniversary of Star Trek™.

Other Editor: Gearhead. "grin"



SST fastRAM Test Flights

Your first Solo...

Okay, we'd now like to kick on the SST RAM. This fastRAM is not initialized until *you* tell it to, because that gives you flexibility, which, as you'll see, is very useful.

There's a program on the SST disk that turns on SST RAM, and in addition, sets its speed: "wait states", "burst wait states", and all that engineering mumbo-jumbo (we'll get into *those* in more detail later). All those numbers are set by the ***name*** of the program, very much like Atari's FOLDRxxx and CACHExxx programs, where, say, naming the program FOLDR200 will add 200 folders to the system folder pool. If you change the ***name*** of the program, you change what it *does*.

This is nice because it's simple to rename a program, but harder to (let's say) edit a configuration file. And, you don't have to answer questions each time you run the program ("What MHz are you using? What Wait States do you want?"). You *may* be running this program every time you start up (like in your AUTO folder), so it had to be done in a way that is "fully automatic".

Pre-Flight Checkout

Editor: Or "user friendly". (See all the jargon you are learning?)

Other Editor: Gee, I thought Fully Automatic was what happened to that hard disk.

First, we need to run the SST RAM initializer in a special test mode that will test SST RAM forever (until you RESET or power OFF). What we want to find out first is if SST RAM is 100%, absolutely reliable. So, we'll run four tests on the SST RAM. In order, they are:

Caution: You must not have more than one program named RAMx.PRG or TSTx.PRG in the same folder. In other words, if you have RAMnbmm.PRG, you cannot have TSTnbmm.PRG, too. If you do and you run the program, you will confuse the SST, because it won't know which parameters to use!

Even if you have a program named RAMDISK.PRG in the same directory, the SST will be unhappy: the RAM initializer looks up its name by looking in the directory for RAM.PRG or TST*.PRG.*

First Test: Basic RAM Check

Write a number into each byte of SST RAM. (Yes, that's 4 or 8 million writes!) Read it back and make sure SST RAM saved it correctly. Then "invert" the number (XOR with $FF if you're a tech-type), store it, and check it. This test makes sure that every "bit" of every byte *works*, both to 1 and 0.

Second Test: Refresh Test

Recheck the inverted values a few seconds after the first test, to make sure that RAM is "persistent". If not refreshed right, the DRAM in the SIMMs will only save values for a short time before they fade away. This test makes sure that the RAM is *retaining* what was written to it, perfectly.

Both these tests are pretty slow tests; they don't stress the memory of the SST too hard speed-wise, but they do test that *every part of every byte* can be used okay.

Then we start entering the speed tests. These force the SST RAM to run at maximum speed and try to make it fail by pushing it hard, typically much harder than you ever will. The speed tests load a program into SST RAM and run it; this forces much faster access to SST RAM. Nevertheless, the SST should pass with no problems on a good SST board with normal RAM. By the way, don't worry; none of these tests can physically damage anything!

Third Test: SST fastRAM Test

In this test, we write a program to all 4 or 8 megabytes of SST RAM, and then we run it! It's not a very intelligent program; all it does is some counting and flashing of the screen to let you know it hasn't crashed. This program should run through all of SST RAM successfully; if there is a crash (2,3,4, or 11 bombs are most significant), then for some reason, SST RAM, while basically working, isn't up to running at "medium speed".

Fourth Test: Burst Mode SST fastRAM Test

In the final test, we run the same program, but run it in "burst mode", which is the fastest possible speed: Mach 3+. This is an extreme stress-test of SST RAM; again, it should pass perfectly. Once again, a crash means that for some reason, SST RAM isn't up to performing at this high a speed.

So, get out your Backup Copy (**the one you were *supposed* to have made before starting the installation**) of the SST Release Disk. Please, oh please, back it up *before* using it on a new system, which an SST amounts to! Copy the RAMnbmm.PRG program, and rename it "TST33mm.PRG". The mm part needs to be changed to your MHz (oscillator value); the copy would be TST3333.PRG for a 33 MHz system, TST3320.PRG for a 20 MHz system, and TST3340.PRG for a 40 MHz system. (I think you're getting the idea.) This "system speed" is basically the speed of the oscillator installed in your SST. The "TST" prefix means you want to run the test version, not the normal "start up SST RAM" program; this is kind of a generic program, which does several things, depending on its name. Be sure to check the READ.ME file on the SST disk for the latest (and greatest) information!

Then go run the copy you just made (by double-clicking on it).

You'll see some normal messages that you will see any time you turn on SST RAM, telling you how much SST RAM you have. I left those in because you need to know how much SST RAM the machine thinks you have, versus how much your wallet purchased! If there's a difference (the SST doesn't think you have as much RAM as you have installed), something is wrong - power OFF and check that all the SIMMs you installed are seated down okay, and are in the right sockets.

Caution: The Shift - Control - Alt - Delete keypress does not work to reset the SST. Be sure to look at the READ.ME file on the SST release disk, for the latest information.

The program will write to every byte of SST RAM, then read it back to test it (this happens every time you *initialize* SST RAM, by the way). Then, because the program you ran started with "TST" instead of "RAM", we'll go forever, doing the testing loop 1-4 above, until you RESET the SST. (Make sure COLDBOOT has been run before you RESET, because you want a Cold Start.) The program with the "RAM" prefix *does not* do the tests 1-4; it just initializes SST RAM.

The SST is capable of running this program for weeks on end; it's designed to. I ran it 12 days once just for fun. In our experience, if the program fails, it usually fails either immediately (first couple of passes) or after the RAM and CPU chips warm up; warmed-up chips have slightly less tolerance for bad behavior than cold chips. I'd watch the program through the first five loops or so, then go away for maybe an hour, and come back and see if it's still running. If the program died, it may have "locked up", have bombs showing, or be back at the desktop; all those things can mean a thermal (heat) failure. A really good test is to leave it on awhile, like overnight, and let the machine get really warm, as warm as it'll ever get.

Note: A computer chip should *never* get "too hot to touch"; mighty warm, yes, but not *that* hot! This is truely a "rule of thumb". The hottest chip on the SST is generally the 68030, because it's usually working the hardest!

Other Editor: No, don't microwave your SST to pre-warm it: while the effect is spectacular, your SST will never work again. Microwaving a CD-ROM (compact disk) is also very spectacular, but may wipe out your microwave's tube.

Editor: Dave learns the most amazing things at HackerCon!

Now, there's another fun test we can run. Power off, count to ten slowly, and power on. (You must wait. Otherwise the Atari won't do a true "cold start" and reset everything properly.)

Boink Test

There's a classic program called "BOINK", which is a bouncing-ball going back and forth across your screen. There's several versions of it on the SST disk, in the BOINK Folder. Let's run it, and make sure that *it* can run in SST RAM. This is not a trivial test of the ST; in fact, it is one of the better diagnostics we have found! It works the sound hardware, the Line-A "Blit" routines, ST video memory, SST memory, and so forth very quickly, while giving you an easy to use indication of how fast things are actually running.

The program FBOINK.PRG on the SST release disk is a SST RAM loading version of Boink; BOINK.PRG is the regular ST RAM loading version of Boink. SST RAM *must be initialized* with the RAMnbmm.PRG *before* any program will run in fastRAM. An "SST" program will just load into ST RAM if you haven't turned on SST RAM. Don't be fooled! (We'll go into exhaustive details in the next chapter.)

Also remember that once you've initialized SST RAM, you may also need to run the DISKxx.PRG, *but only if your hard disk software requires it.*

Pre-Flight Checkout

First, rename the TST program to RAM33mm.PRG, with the MHz value of your oscillator for "mm". Then initialize SST RAM with the RAM33mm.PRG program by running it. Now, try running the original BOINK, with the cache off (turn the cache off from the OPTIONS menu; it is OFF when there is *no checkmark* by it). Watch a moment to see the speed, then "Quit" from the "File" menu.

That's essentially "normal" ST speed BOINK. Turn the cache ON (to where the checkmark shows). Run BOINK again. You'll see about a 4 times improvement in speed.

Now, cross your fingers, which does make it harder to type, admittedly, and then run FBOINK. FBOINK is set up to load into SST RAM and run in burst mode (warp drive).

Editor: Afterburners!

"FAST"Boink should really fly along, around 8 to 12 times faster than Boink.

If you get a crash when trying to run FBOINK, you're probably having interface problems or software problems of another sort, *particularly* if you passed the TSTnbmm.PRG diagnostics, which tests SST RAM. Problems could also include your hard disk; it may not know about dealing with SST RAM, for example.

Did you remember to run the DISKxx program (but only if it is required by your hard disk software)? Check the list at the end of the "Hard Disks, Floppies, and the SST" chapter to make sure.

The TST diagnostics pretty much exclusively test the SST part of the machine; FBOINK tests out the SST to ST socket, wiring and chips, basically the "interface" between the ST and SST.

Most often, a problem here is caused by the SST board not being in the socket firmly, or by the board tilting towards the SIMMs; it's sorta easy to press down on the SIMMS and get the board tilted, lifting 32 pins (half) out of the 68000 socket. Try pressing down *hard* on the chips right above the 68000 socket *with the power off*; if you feel any wiggle, you probably just fixed it.

No fix? You may want to reseat the ST's chips; in my practical experience, anytime I take apart an ST, it seems to wiggle it enough to where I need to reseat the socketed chips.

No fix? FBOINK is loading the program into SST RAM all right (well...maybe), but can't run from it (or what was loaded in was corrupted along the way). Check that the SIMMs are seated down, and if you're running 8 megabytes, perhaps try 4; you may be stretching your power supply's luck. The entire ST & SST setup pulls around 2.3 amps at maximum load, and the Atari supply is rated at 3.0 amps, but perhaps your supply is getting overloaded. *Some* Atari power supplies don't deliver the rated 3 amps, we are told!

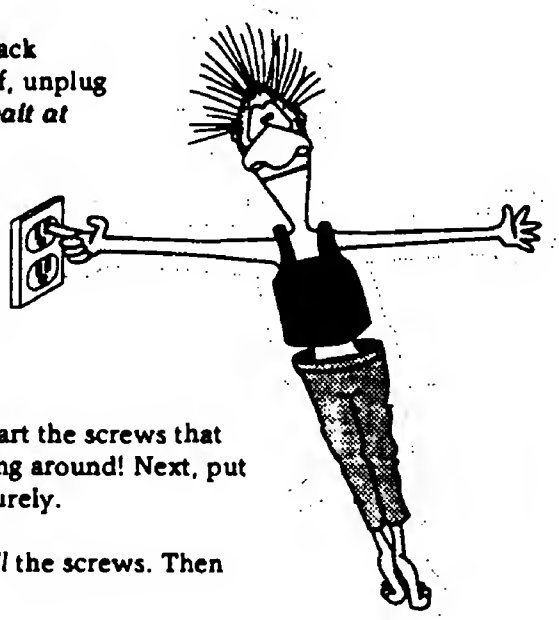
A definite overload condition is happening if your whole ST shuts down; this is called "thermal shutdown". It means the power supply got too hot, figured there was a problem, and shut itself off before it burned out. The only cure is to wait until it cools.

Another definite known possible problem are the capacitors in the Mega ST power supply; they tend to be weak. If your ST needs to warm up a bit (particularly, with the fan going "tick-tick-tick") before starting up, replace the power supply capacitors, and really heat up all the solder joints in there; I've seen many of them crudded up and boiling with rosin, a very bad connection. Boil 'em out.

Still down? Give us a ring and we'll give some advice, and help you out. The SST and ST combination is extremely complex, and there are a lot of factors which can adversely affect it. Most of them are easy fixes, once we figure out what's going on, so don't worry.

I would personally let FBOINK run all night, or a few days, whatever you feel like (maybe after work). It is a very good test of the machine. Also, if you ever suddenly develop erratic crashes, FBOINK can tell you really quickly if the SST/ST interface is running reliably.

Reassemble Your Mega



Okay, your SST is working; now you need to put your Mega back together, and get all those screws back where they belong! First off, unplug everything, and let the power supply capacitors discharge again; ***wait at least 10 minutes; you really don't want to get "zapped".***

At this point, your bottom shield, ST motherboard with SST attached, floppy drive, and power supply should all be in the bottom half of the Mega case, and hooked up. At this time you should also install any *other* devices you want inside your Mega (like a video board). If you add another device, be sure to retest everything again *before* you finish reassembling it.

Make sure the power supply is securely fastened down, and start the screws that hold down the floppy drive; you don't want either of them bouncing around! Next, put the top shield on, and bend all those little tabs to hold it down securely.

Put the top of the Mega case on, and then install and tighten *all* the screws. Then reconnect all those cables (again) and you're ready to fly.

Well, you've got the SST basically running. But, there's a big difference between basically running and running at maximum possible speed; we need to tune-up the SST so that instead of Warp 4, we can be hitting Warp 9. That's covered in the next chapter.

To shut off the SST, just power off that same way you've always done it. Be sure to replace the ejection seat arming clips, set magneto switch to "BOTH OFF", pull throttle back to "IDLE/CUTOFF", and check that the brakes are on and locked. Then open the canopy and climb out. (It's also a good idea to make sure the flight ramp has been wheeled up and locked - it's 18 feet down!)

Editor: I hate to interrupt the flow here, but I noticed if I added up all the "run overnight to test" stuff, the SST installation and pre flight check takes about a week. I don't really think these people are going to wait that long, do you?!

*Other Editor: No, odds are the SST will work fine the first time, if they **read the manual**, and **follow the instructions.***

Jenny and the Snowman

So I had this UNIX computer on loan to me. It was sitting there doing absolutely nothing 20 million times per second. It had a 20 MHz 68030 and floating-point hardware in it, plus 8 megabytes of RAM and a fast hard disk and three, count 'em, three serial ports.

And I mean it did NOTHING. Every now and then I'd switch the ST to talk to it (we have a system of computer intercabling which works wonderfully every time I go completely through it, tracing every cable, to find out where it goes, remove the obsolete labels, and put on new ones), and ask it if it was still awake. It would say:

"$"

Which meant it was still awake. Wooooo. Thrills.

Just then, fate intervened. I got a "ray tracing" program. These programs are neat; let me tell you about them briefly.

A ray tracer is a way for a computer to build up an image of something that only exists in the computer's and the programmer/artist's imagination. For instance, we could have a floor made of bricks, lots of colored spheres floating above the floor, and two walls that are mirrors. There would be a few lights above all this. If this scene really existed, you would "see", through what's called a "peephole", the reflective walls showing you a view of the floor's pattern, and more interestingly, views of the other spheres, if they were in view.

Well, in a ray tracer, the computer builds up an image to be displayed on the monitor of what the viewed area (in this case, the corner of a room) looks like. Again, you'd see the floor, the

spheres that your viewing angle lets you see, and so forth. The lights would provide illumination, and the spheres, of course, would have to cast shadows on the floor below them for realism. There are many details, like how bright the lights are, what color(s) and texture(s) the objects are, and so on.

Now how it does this is unique; there are many other ways for computers to build up an image (computer word for it: "rendering"). (Funny, I always thought "rendering" was when you boiled something in water down to its parts... guess I've been reading too many Black Company books, by Glen Cook.) (Excellent Stuff!!). But other techniques involving "shading", "wireframing", "hidden line removal", and 97 other buzzwords exist in computer land.

Ray tracers take one, individual beam of light, and follow it from the lamp through the path it takes to the peephole, where we'll see it as a screen image. They then take the next ray, the next, and so forth. It takes a long time because each light generates a lot of individual

rays, and each one has to be traced out, and some of the tracing is involved mathematics, real heavy stuff.

As an example, let's say we have one blue sphere in there. The light beam starts out at the lightbulb it is generated from, and travels from there to the blue sphere. (The computer has to figure out that it doesn't hit anything else first, which is not so easy in a 3-D area.) When it hits the blue sphere, the "white" light coming off the light (assuming a white light) becomes blue; technically, the blue sphere absorbs everything but the blue component of the light ray. The beam then bounces, which is an interesting math function of the incoming angle of the ray and the diameter of the sphere. Presumably it bounces to the peephole. You then have to figure out just where in the peephole it comes out. Typically, peepholes are set up to the size of computer screens, dot for dot.

Now, that was idealistic ray tracing, just to get you familiar with the concept of walking alongside light rays. The real world of ray tracing is far different. For instance, you don't trace out every light ray, because most light rays don't ever get back to the peephole! They get soaked up in the floor or something. So what you do is take every dot of the peephole, and sort-of "reverse trace" them, finding everything that's visible to that dot and seeing how all of them add up in terms of color, brightness, and so forth.

Say "Ray Tracing" to a computer hacker, and you'll get an immediate response: "Takes a ridiculous amount of computer time". And it does! Even today, with awesomely powerful computers like Crays, it can take hours to do a ray trace; there are just so many light rays to trace that it is an enormous job for the computer. This is why Ray Tracing isn't used that much; while it can give spectacular results, the technique itself isn't foolproof by any means, other techniques work much faster and have gotten more foolproofed, and no one has the patience to wait (say) 60 hours for a Ray Trace to complete, while it is completely tying up a computer!

So Jennifer, who is 8 years old and fully computer literate, was looking at this incredibly boring ray trace I'd done, on the useless UNIX computer. I think I had taken a picture, so to speak, of a red, green, and blue sphere floating in the corner of a brick room. Wow. National Endowment for the Arts stuff (NEA) stuff!

And I said to Jenny, "Got any ideas for a picture?"

She eyed the spheres, and promptly said, "How about a snowman?"

Genius! (That means, it was something I could figure out how to do.)

So we sat down together and sketched it out. I introduced Jenny (1st grader at this time) to the X-Y co-ordinate system, and she picked it up as naturally as breathing. We designed a Snowman, with 3 spheres to him (Red, Green, and Blue, which are primary colors and really stand out), two eyes, and a nose (reflective). Then we designed a SnowDog, just for fun, which was much more complex. Jennifer was rattling off X and Y co-ordinates and sphere radius and colors to me for some time.

So I set up the trace, "told" the computer to limit itself to 4 hours of real time (thus, it didn't "do" every dot; it guessed on a bunch of them), and we came downstairs awhile later, saw a crude sketch of our drawing, fixed some things up, did another 4-hour check – and I told the computer to go for it. Do a perfect trace.

Well. I started the thing up at 8 P.M., at the kid's bedtime; I went downstairs the next morning at 9 to see the completed drawing. It had completed around 5 in the morning, it said. This was the first chilling note I had of what was to come; remember, this was the fastest computer I had ever worked on here (this was way before SST). I pulled up the image, and it looked pretty good. The SnowMan and SnowDog were drifting in midair, so to speak (I had not specified a floor!), so I added a floor, and just for the heck of it, I added a mirror "behind" the SnowPair, so that I would see them and their reflection. Again, I set the computer to running.

I mean, as far as I was concerned, this was great. I was getting some use out of this UNIX

machine that otherwise was sitting there, might-as-well-be-off. And, I must confess, I was impressing my daughter on "our" project.

... dads need that...

Jennifer came by to see "her" snowman after school, and I proudly showed her the image... whups. I'd put the "mirror" right in front of me, so to speak, and it wasn't a "one-way" mirror; since eyes don't give off light, I got a black screen for all the hours of number crunching. She gave me one of those "It's okay, Dad, don't be so hard on yourself; you're old..." looks, as I explained how important it was to put a minus sign on the X co-ordinate, and we re-ran.

Well, there it was the next day. I had missed one X co-ordinate defining a rectangle for the mirror, and the mirror neatly sliced through the SnowDog like a Ginsu knife. I did not show this image to Jennifer, imagining her comments. Probably this was a bad idea; my imaginings were probably far worse than anything she would have said.

"Nice boo-boo, Dad."

"Boy are these computers fun." in an absolutely flat voice.

"Let's play Nintendo, Eric!"

Next day, she was very pleased with her snowman, in its radiant red/green/blue, the snowdog in all its colors (even if its tail wasn't perfect yet), and most of all their reflections in the mirror. The mirror was 100% reflective, and there was no sign there was a mirror there, so to speak (no mirror "frame" or anything like that.) So what you "saw" were two Snowmen and two SnowDogs.

And I got to thinking of the time she and Eric had pointed the video camera at the TV it was displaying on, and gotten the "hall of mirrors" effect (similar to holding two mirrors apart and looking between them; you get mirrors stretching to infinity.) And I chuckled evilly to myself, and added a box of mirrors around the snowman, four of them in a square. I carefully made sure my "viewing eye", or "camera", was inside the mirrors, and turned it loose.

The machine crashed a few hours later. It let me know it had gotten an inch or so down into the image when it crashed.

Now, I mean, I hadn't written the program, and I had next to no chance of debugging it. I wasn't even sure I had set it up correctly for UNIX; I am no UNIX expert!

I'm embarrassed to tell you how long it took me to find such a simple conceptual bug. Let us just say that after a certain amount of time measured in days, it occurred to me: "What happens to a light ray bouncing between perfect mirrors?"

Why, it bounces forever, Dave. And eventually the machine runs out of room to keep track of all the bounces and gives an out-of-memory error.

So, I set the mirrors to 95% perfect instead of 100% perfect, and re-ran the program, stripping out all the "fluff" I'd put in there. (They would have slowed it down).

Sixty hours later, the ray trace was done.

I could see from the little display that the computer was slowly, painfully slowly, making progress, moving from dot to dot, summing up all the light rays that impacted on that dot. I am not used to 20 MHz 68030 machines going this slow, nor UNIX! This had always been a pretty zippy machine. (Now, with SST, you know that 20 MHz is not killer speed – I didn't know this back then). So I let it run and hoped it was doing something constructive. Later on, I figured out I could have looked at the image as it was part-way built up and seen if it was working, but I was too new on the machine for that.

60 hours, of course, is two and a half days. It seemed like weeks. Each day, Jenny would bounce downstairs (it reminds me a little of Tigger), and ask to see her SnowMan and SnowDog. To my embarassment, under the withering eye of my daughter, I could only say "The computer is still working on it." I believe her opinion of computers suffered a lifetime loss from this experience ("Yeah, my Dad works with computers. Wow, they're slow.")

Third day arrived. Jenny came downstairs, asked to see the picture. With positive dread in me – for I remembered all too well the goofups I had made of so many hours worth of work – I pulled the image up.

And glory be, the computer had mercy on me. What we got was a thoroughly spectacular image of hundreds, maybe thousands of Snowmen and Snowdogs, with beautiful-living-color. Jennifer gasped, said, "Wooooo!!".

Jenny and the Snowman

And I had one of those rare, rare moments where I feel fulfilled as a Dad. I had impressed my daughter and done something with her. We did black & white prints on the LaserWriter and showed them off. Everyone was impressed; I did notice out of the corner of my eye that perhaps Sandy was impressed in the way you must be impressed by what your kids bring home from school...

Probably as a direct result of that, I became more or less snowman-obsessed. What I did was run **hundreds of traces** of the SnowMan epic, and I had one mirror of one wall open and close something like a big swinging door. This caused all the reflections to move fairly radically. Since I did it in slow increments, if you played one picture after another, it was (Wow!) animated! (I'm sure Disney rolled in his grave.)

This process took **months**. What I would do is set up the Tracer instructions, and instruct UNIX to run one trace after another after another. When they were done, I'd download them to my Mac II and look at them. (I got a little expert help making a Mac II displayer from Dan Moore, who did the Spectre Menus – that's tricky). Took me a few minutes a week to set up the next week's work. I learned a great deal about UNIX during those months, as I'd do dumb things like fill the disk up until it crashed or whatnot. Ultimately, I hope that this knowledge will somehow serve me, but as time goes by, I hold my breath less and less.

Now sure, my Snowman Animation takes up 25 Megabytes of disk space that I still haven't reclaimed. And sure, I have showed it to Jenny, Sandy, Eric, Jamie, my Mom and Dad, both brothers, their wives, their kids, visitors, the mailman, the milkman, a process server, the cat and dog... I mean, I'm proud of my little ray trace project. (You guessed?)

Jenny and I had a great time with the SnowMan, and she has always enjoyed talking about how far her idea went. She's gone on to do other things with computers that she loves, from playing Manhole™ to Cosmic Osmo™ to Spaceship Warlock™.

From making a floppy disk sandwich, to using PixelPaint™'s super-drawing modes, to fearlessly plugging in circuit boards for testing (yes, we pay 'em!), Eric, Jenny and Jamie are about as afraid of computers as you are afraid of a toaster.

And Jamie, the 3 year old? A few hours ago, he was driving my Mac IIfx, playing "Cosmic Osmo", a wonderful game, using just the mouse. And about every two minutes, "Come see, Daddy!" He's already found out how to look through the telescope and steer the spaceship... (Which I never figured out!) So I'll leave you with a tiny moral that I found out accidentally.

It's easy to say my household is "different" because we are in the computer business. But

that's not really relevant. Lots and lots of people have taken home old IBM PC's, for instance, with color monitors.

• Your kids will be working with computers all of their lives. You might as well get them used to it now, and with any luck, get them past being afraid of the machine, and maybe even into learning a little about them. (There's always going to be a demand for programmers). And perhaps the only way to get them to teach themselves (for they will learn far more than you or I given the will to do so) is to make it fun for them. Jenny had fun with the Snowman and moved on to having lots of fun with other things on the computer. So has Eric, and Jamie; they damn near self-start with some of the computer games.

• The other thing I now know is that computers have advanced enough to where people who aren't going to become computer experts need to be using them, as a matter of speed and convenience. The era of computers for computer's sake is ending; I was lucky enough to see the start of the microcomputer part of it, and have an interesting time during it. Now it's time to really make computers easy enough for people to use, for those of us who aren't willing to put up with little computerisms; now it's time to make them as easy to use as a toaster. We finally have the memory, the CPU speed, the programming techniques to make the computers come several significant steps closer to people, instead of making people come to them. And, heck, if their experiences with previous computers haven't scarred them too badly, maybe they just will.

The SST is a step along these lines; I believe you will find it makes your computer easier to use by making the ST less limiting and by opening up new ways of doing things. Heck, if you can run a couple of 2 Megabyte RAMdisks in SST RAM, why store temporary files on disk? Or maybe it'll give you the temporary space to use for extra tries at drawings, sound captures, or image captures, without worrying about running out of disk space.

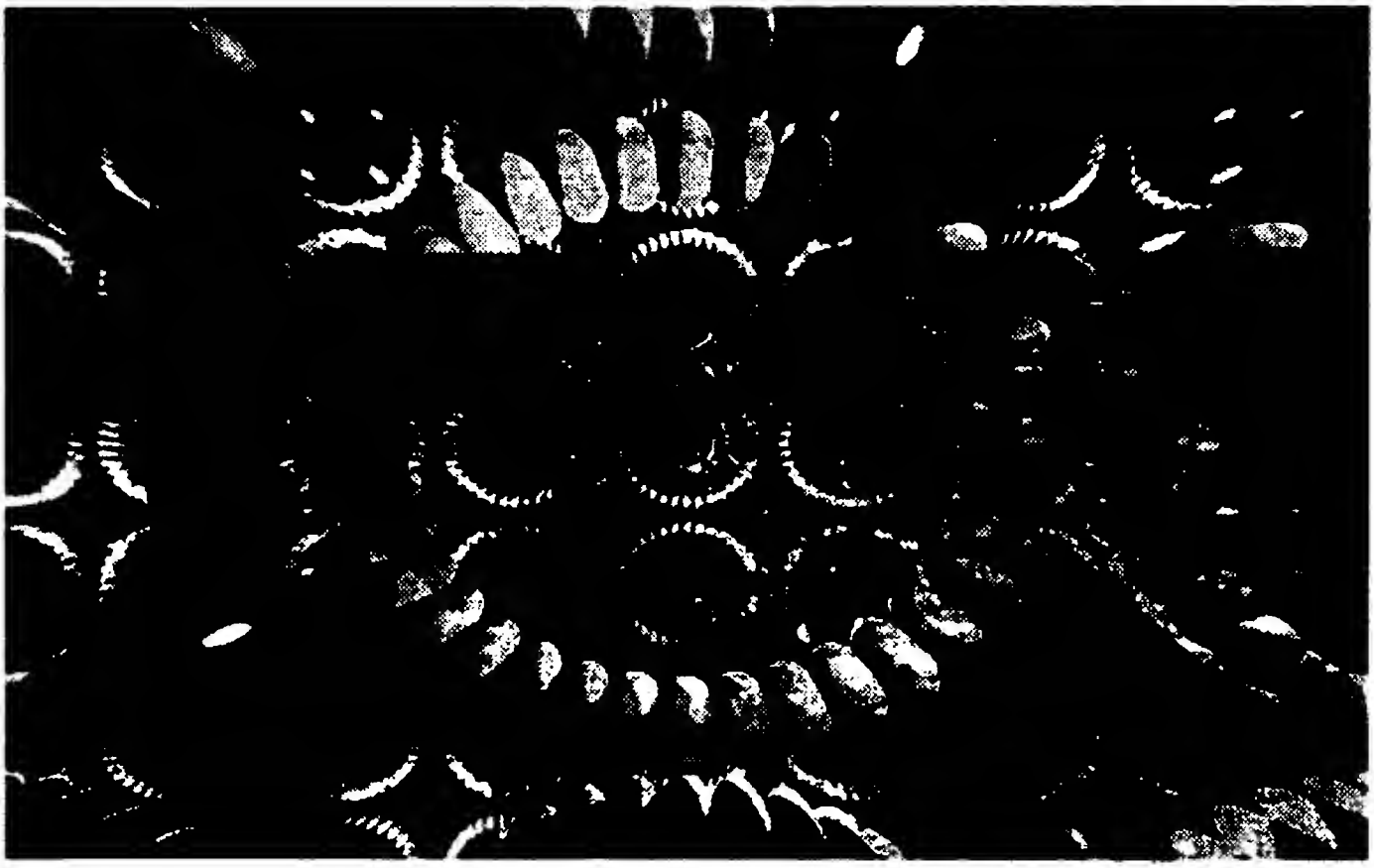
Yes, there's a ways to go, but we're on the right track.

See, Jenny taught me a lot by being enough of an artist to dream up making a Snowman, using that ray tracer I showed her. For without her as an artist, dreaming up the Snowman idea, all that expensive UNIX stuff was useless. Just as without something to write, a word processor (like the one I am using now) is useless; without something to design, a CAD program is worthless.

Now is the time we need the dreamers to find computers, and create, the most.

Jenny and the Snowman... I think about it a lot.

Jenny and the Snowman



Test Flights

As with any aircraft, you have to do test flights to push the hardware to the limit, in order to get the maximum performance. Your SST is no different; you should test *your* configuration in order to get the most out of it.

If you have the RAMnbmm program in your AUTO folder, and do not want to initialize SST RAM when you start up, just press either "SHIFT" key while booting.

At this point, your SST is basically running. But we're slowing it down quite a bit in ways that *may* not be necessary (especially if you're running at less than 22 or so MHz). Let's tune it for maximum speed, while still remaining reliable. We do this by "trimming off" unnecessary wait states.

Running 3 & 3 wait states, which we were doing during basic testing, ***is not maxxing out*** the performance of the SST RAM. We've given you all the flexibility you want, which means you can push the SST RAM chips possibly faster than they can go. How fast you can go really depends on your **68030, your** chosen oscillator value (your MHz), *and* your ST hardware. This lets you fine tune your SST to the very limit of what it can do reliably on your ST.

The RAM Initializer/Tester (**RAMnbmm**.PRG) lets you set the wait states, burst mode wait states, and the MHz you are running at, just by changing the name you give it.

There is a restriction or two here to pay for this convenience. The RAM tester figures out its set up by looking up its name in the disk directory. This means if you have more than one program named **RAMnbmm.PRG** *or* TSTnbmm.PRG (where "nbmm" is ***anything*** you set it as), the RAM Initializer/Tester won't know which one to use! Please "turn off" any RAM initializers and testers you're not using. The easiest way is to click on the program, press "S" to bring up its "Show Info" box, then press X to change the name to .PRX instead of .PRG. That way, when you run the program and it searches for RAMnbmm.PRG in the directory, it will ignore all the programs with the name change.

Should you forget, the program will remind you; it has to. Even if you have a "RAMDISK.PRG" file in your folder, the RAM Initializer will get very confused, thinking it should set the MHZ to "SK", for example.

Wait States

By now you're asking, What Are Wait States, Anyway? It's a term that computer macho-types use. It describes how many "machine cycles" (those 33 million a second thingos that your oscillator generates, or *whatever frequency you run at*) that your RAM is allowed to get information, once asked for it.

Editor: And here I always thought it was when you were sitting on the runway waiting for takeoff clearance from the Control Tower!

The 68030 has two different kinds of wait states: normal and burst mode, which we'll get to in a bit. Normal wait states work like this:

The 68030 says, "Hey, RAM, I need 4 bytes starting at location 10,000. Go fetch it and put it into my IN basket!"

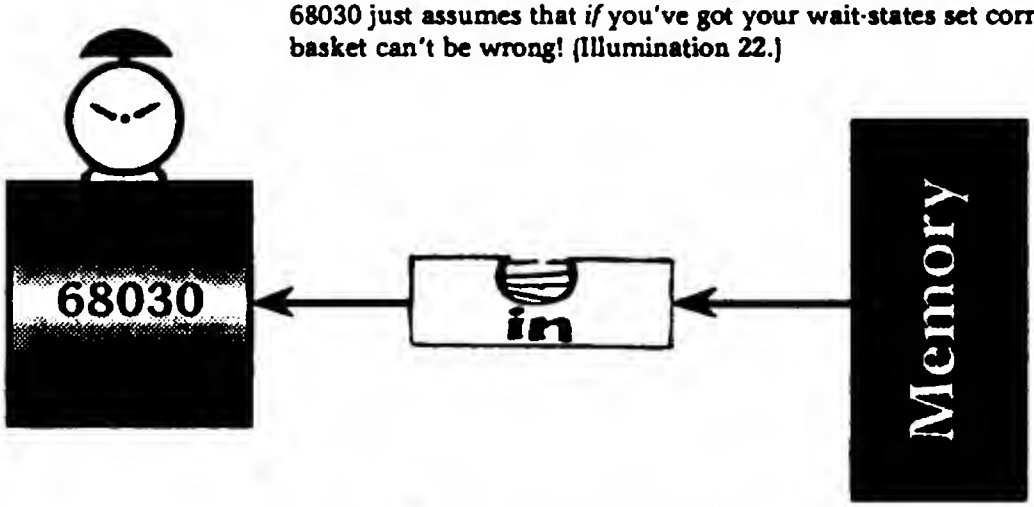
RAM replies, "Okay, I'm working on it. I'm putting you on HOLD." RAM pushes the HOLD button, and the 68030 starts listening to Muzak™, just like you and I do. The 68030 is "Wait Stated", technically. It irritates the 68030 about as much as it does us when we get on HOLD! (*I think the worst ones are the ones with commercials...*)

The 68030's patience begins to disappear as it counts away wait states (from one through four), a tick at a time.

RAM in the meantime is frantically shuffling through the files, its desk, under the table, where-ever, looking for the data in location 10,000. Really, it's a speed race; it takes a certain number of nanoseconds (billionths of a second) for RAM to get the value out of storage and dump it into the 68030's IN basket.

Editor: Remember, a nanosecond is less than a foot (30 cm) long – it's the distance light travels in one billionth of a second.

After the number of wait-states you specify, the 68030 loses its patience, and says, "Gimme what you got now, Bud!". It grabs, or "fetches" (in programmer talk) whatever is in the IN basket at that moment, without ever really checking that the data is okay; the 68030 just assumes that *if* you've got your wait-states set correctly, the stuff in the IN basket can't be wrong! (Illumination 22.)



Illumination 22 - A Normal Wait State

The RAM hands over something, be it the correct value (most often), a distorted value (say, a bit or two is wrong), or if RAM wasn't quite done, a complete "flat-line" (as we call it), which is $FFFF FFFF hexadecimal (more programmer talk; it is some *big number* in decimal); this means that the RAM didn't get the information out of itself in time, and gave absolutely nothing to the 68030. In this case, resistors in the circuit "pull up" all the lines to logic "1", and the 68030 gets the highest possible number, which programmers write as $FFFF FFFF (An "F" in hexadecimal actually represents four bits on: "1111" in binary.)

Editor: So "FFFF FFFF" in hex is "1111 1111 1111 1111 1111 1111 1111 1111" in binary, and "4,294,967,040" in decimal.

If you get a "flat line", you will know, because you'll get 11 bombs onscreen. This is a "Line F" trap (even *more* programmer talk), and means that the 68030 just tried to execute an instruction that was (guess) $FFFF FFFF! The first F is a special code to the 68030 meaning a "Line F Instruction", but the rest of the instruction is illegal, so the 68030 bombs out to tell you something is very wrong. This particular error usually means you're just over-running your RAM's speed. (It can also mean your RAM is not working – you get the *same* error when trying to read from nonexistent *or* non-working memory.)

If you get a bit or two flipped (distorted information), then you'll generally crash a different way: 2, 3, or 4 bombs. The solution: ease off the gas a little, and add a wait state, which gives the RAM another "tick" to get the correct data to the 68030's IN basket.

So what we do when tuning (running test flights) is to ensure RAM has enough time to fill the IN box reliably, but not so much time that the data is sitting in the IN box gathering dust. Believe me, it matters *a lot*; you can't just set maximum wait states and expect to go Warp 9! Also, since the SST is a complex piece of hardware, several factors come into play that we have to take into account while doing our test flights.

Oscillator and CPU Speed

Obviously, the faster your oscillator, the faster you are pushing the board (16, 25, 33, 40 MHz), *and* the faster RAM has to work to get a correct value back by the time the 68030 grabs it. In particular, up around 40 MHz, you may need all the wait states you can get, and if you use particularly slow RAM chips, they may *still* not be enough. On the other hand, at slow speeds, you can kill off just about all wait states.

SST RAM Speed

The faster a RAM chip you have, the more likely it is that the RAM chip can grab the data and get it to the 68030 in time. RAM chips are rated in nanoseconds, which is billionths of a second! ST's commonly use 120 nanoseconds; the SST uses 60, 70, or 80 nanosecond chips; we don't recommend slower ones (numbers over 80). 60 nanoseconds is about the fastest RAM you can find for a reasonable price. (I have heard rumors of 55 ns SIMMs, but never seen any.) Our recommendation is that you use 1x8 SIMMs at 70 ns; remember they *must be* "page mode", or the SST won't work.

SIMM prices are positively guaranteed to fluctuate. I have heard rumors that since the 1 Meg SIMM market is flooded, factories are switching to 4 meg production, and 1 meg prices may move back up as production slows down. We'll see. (No, you *cannot* use 4 meg SIMMs in the SST; that would require a *major* design change!)

Then, there's the fact that very often, RAM chips just plain run faster than specified. We very carefully made a chart of what wait states were needed at what MHz and what speed RAM, then tested it; the chart didn't help us much. The RAM chips often worked where the equations clearly said they shouldn't! It could well be that we got a batch of particularly good RAM chips; it could be the plant that manufactures them makes just one type of RAM, and stamps different speed ratings on them despite their true speed; or it could be good karma from a previous life. We don't know; we just see and take note of the effect. (We think the Karma theory is unlikely, though). Many boards run just fine in this "No Man's Land" of low wait states, where RAM *theoretically* shouldn't be able to keep up; right now, downstairs, I have an SST that has been running a test program for over a week, day and night, at very high MHz, with fairly slow (80 ns) RAM. It shouldn't work, but it works fine.

It was the closest thing I had to garlic!

We keep garlic around it and hold up crosses whenever we get near (just as a sensible precaution) should it decide to come to life as well.

Finally, we have to take into account the tolerances between individual SST boards. Electricity itself moves about one foot in one nanosecond; many of the chips on this board had to be special, high speed versions, to get the electricity through them in time to make everything work! 33 to 40 MHz is terrifically hard to keep running reliably; up to forty million things are happening each second, each one of them just aching to go wrong. The high speed chips have their own quirks; some are 1-5 nanoseconds faster than others, even out of the same package! So it is possible that one SST board will require one more wait-state than another when running in "No Man's Land", even if they have "identically rated" chips.

This applies only to "No Man's Land" (low wait states); we do guarantee the boards

will work to rated spec. We're just being honest enough to tell you that very often, they can do a heck of a lot more, and you should try to take advantage of it!

So the approach we decided on is not to force you into any particular wait state configuration; it's to let you tune your SST board to your ST, with whatever RAM and speed you use. The tuning program is the same RAM tester we just used. What you do is start "turning down" the wait states, and keep running the RAM tester. Sooner or later, you will over-run your RAM's ability to feed that 68030 IN-box, and the tester will crash; *it is supposed to work this way*, to let you know that you've reached the wait state point where SST RAM is unstable.

Yes, I did think about intercepting the crashes, and presenting a nice, neat message like, "You are pushing SST RAM too hard; back off a wait state." Problem is, other software damage is often done by a crash that prevents such a message from working; about a quarter of the time, the message didn't show, despite my best efforts. It's better to just let you know the program is written to stress the SST RAM as much as possible, and if it crashes in any way, you are pushing too hard, and need to add a wait state.

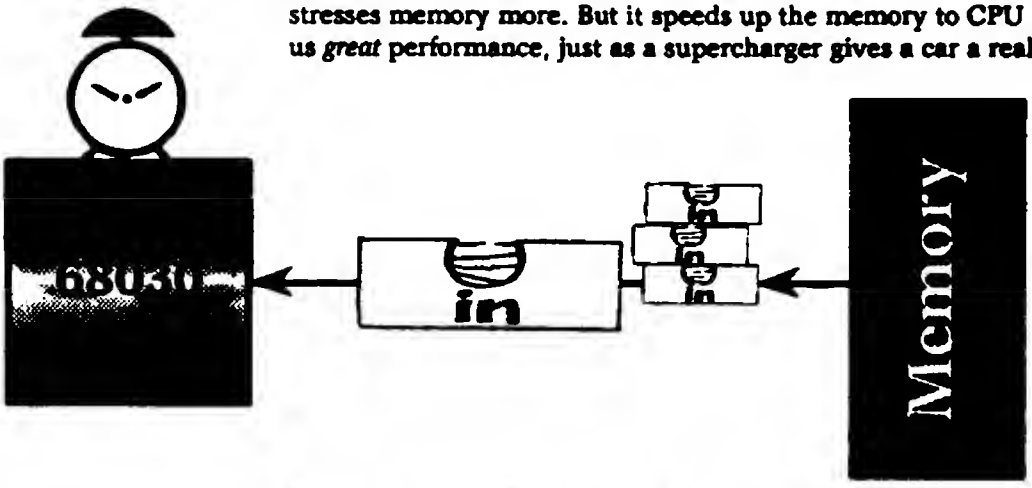
While I hesitate to release a program I *know* will crash, this is just the *most* practical and effective way to tune in your SST. You see, a RAM being run a bit too fast will fail only once every few billion accesses. All the other RAM accesses will be fine. This translates roughly into a crash every 10 minutes, maybe. What makes detecting marginal wait states even harder is that SST RAM is not always being used. So it's best to run TSTnbmm.PRG to stress SST RAM for awhile to check for marginal wait states. And it often only really fails after warming up!

Editor: Remember that phrase, "pushing the envelope"?!

Burst Mode Wait States

I've already explained what a regular wait state is; now I need to explain a specialized type of wait state called a Burst Mode Wait State.

When a program is running in SST RAM, and burst mode is enabled, the 68030 turns on its afterburners and starts "supercharging" its intake of instructions. It works like this. Instead of going to memory for an instruction, doing what the instruction says, going to memory again, and so on as usual, the 68030 tells memory to give it 16 bytes (which is up to eight 68030 instructions!), all in about 5 machine cycles. This is incredibly unbelievably far faster than those instructions would normally be read in, and correspondingly, it stresses memory more. But it speeds up the memory to CPU intake by far, and that gives us *great* performance, just as a supercharger gives a car a real improvement.



Illumination 23 - A Burst Mode Wait State

Note: Anytime the CACHE entry on the menus is on, burst mode is possible; it only kicks in on SST RAM, since ST RAM isn't wired for it.

Editor: Afterburners, Dave. Afterburners! Like in the movie Top Gun.

The 68030 must do four "fetches" of four 32-bit "longs" from memory, to complete one burst mode grab, for 16 bytes (4 times 4) total. The first fetch, the 68030 does a normal (sort of) wait-stated grab, but it also warns memory that it's got a burst mode fetch series to do. Then, the 68030 goes wham-wham-wham, and grabs the next three sets

of 4 "longs" that memory hands it, real fast. In fact, memory itself must figure out where the next bytes come from; the 68030 assumes your memory system is smart enough to calculate that! (Ours is.)

So you see, a supercharger (or afterburner) analogy is very close; memory is "accelerating in" bytes to the 68030. A burst mode wait state is how long the 68030 waits before grabbing in those additional three fast grabs, and you can bet it's *fast stuff.*

So, burst mode is actually a mix of 1 normal fetch with a normal wait state (handled just like a regular wait state), and 3 special, unusually fast fetches with the burst wait state. (Illumination 23, previous page.)

Big Time Really Super Neat Thing to Know

Now if you cut normal wait states by 1, you can speed up the SST some, true. Every normal SST memory access comes in faster (or, at least, the 68030 thinks so; whether or not your memory can do it is another question). But, if you can manage to cut burst mode wait states by just 1, you've in effect cut *three* wait states off the total for that burst, because they're done three times! So it's really cool to trim down burst mode; you get three for the price of one! This really kicks the SST into overdrive.

Editor: Afterburners. "sigh" Dave's mind must be on his Camaro again.

Other Editor: Superchargers! It's a better analogy/metaphor.

Editor: Want to do the dishes for the rest of your life, Oh Mystic Writer?

Other Editor: Afterburners...

If you're curious, the 68030 then sets those 16 bytes down inside itself in the "instruction cache", and reads what to do from them. Provided your program doesn't jump around, it pretty much plows straight through them, at very high speed. It's all internal to the 68030; there's nothing to slow it down. When it gets time, the "68030 pipeliner" sees that the CPU is getting near to running out of instructions, and another burst mode fetch is done. The "pipeliner" tries to make sure that the 68030 never waits - and it does an excellent job (which is why the 68030 is the heart of many high power work stations).

RAMnbmm.PRG Specifications

Now that you know about wait states, and why it's so good to have as few as possible, let's begin the test flights; in other words, it's time to actually fine tune your SST. What we'll do first is tune the burst mode wait states, then we'll tune the regular mode wait states. In our experience, burst mode waits are easy to tune (if you go too far, you'll crash nearly immediately); regular waits take a little longer to be sure of.

Remember our RAM test program, TST33mm.PRG? We currently have it set up as RAM33mm.PRG (from when you initialized fastRAM to run the FBOINK program), where "mm" is your particular MHz value; for instance, a 40 MHz SST would be TST3340.PRG, and a 20 MHz SST would be TST3320.PRG.

Remember, the SST takes it's operating specifications from the *name* of the program. If the name is TSTnbmmD.PRG, then the SST runs an infinite loop cycle of fastRAM tests (which you have to RESET out of); adding "D" is a special option. If the program is named RAMnbmm.PRG, then it initializes the SST fastRAM, and away you go!

So, what does the "nbmmD" part stand for? Well, it works like this:

- "n" is the number of normal wait states, from 1 to 4. You can't go to 0 in the SST hardware, but don't worry about it; with SIMMs, you can't get zero wait-states, so you're not missing out anyway. The default starting

place is 3; if you want to try running at 40 MHz, you should start out at 4.

- "b" is the number of burst mode wait states, from 0 to 3. You *can* get 0 burst waits, but it usually takes a miracle (and really fast SST RAM). A good starting place is 3.

- "mm" is the MHz you're running at. That's straight off the oscillator plugged in the board. The SST needs to know this to set up the memory refresh speed appropriately. In fact, this is not a critical number, as long as it is "in the ballpark"; we are *very conservative* on our memory design (which you will appreciate, if you've ever had experience with a flaky one), and memory itself has a lot of refresh caution built in these days, too. This value can be from 12 to 44 MHz on the software end; but hardware realities require us to run slightly over 18 MHz to ensure synchronizing with the ST, and slightly under where ever *your* ST & SST maxxes out.

  Technical Note: The SST *must* run slightly faster that 16 MHz, so the GAL logic can grab signals *before* the ST can get to them. If you use a 16 MHz oscillator, *the SST will not work!*

- "D" is an option only for TSTnbmmD.PRG, which forces the program to run the burst mode test forever (test #4 from the previous chapter, where a program is loaded into SST RAM, then run in burst mode). This is particularly useful for tuning burst mode, since you don't *need* to check other RAM functions while burst mode tuning.

The letter "D" stands for "D"ave, not for anything like "loop" or "do it forever". This is how programmers build back-doors into programs; there are other ones in this program, too. (None, however, are real interesting; they are other tests I needed at one point or another in the hardware/software debugging process).

Editor: And besides, they're pretty much "commented out" in the release software.

So, you can see from the current name, RAM33mm.PRG, that we are not set up for burst mode test looping (no "TST" or "D" there) and that we are set at 3 normal and 3 wait states. Time to change the name.

Tuning Burst Mode Wait States

Subtracting 1 from the burst waits gives us 2. (3-1=2. Easy, huh?)

So, click on the program name (just once, to highlight it; twice would run the program). Press S. This will bring up the "Show Info" display on that program, which lets you change its name. Change the "RAM" to "TST" and the burst mode "3" to "2", preferably using the arrow key to move left, BACKSPACE to delete the 3, "2" to insert the 2, and you're done. Now also add a "D" to the 8th character of the name. For example, if you're running at 40 MHz the name should be TST3240D.PRG: 3 normal wait states, 2 burst wait states, 40 MHz, infinite burst loop test.

You'll soon get the hang of quickly editing program names. The way I just described works the best. Believe me, if *you* had changed the name 7,000 times (*like I have*), you'd know the fastest way, too!

Run it. You'll see it cycle through the normal tests once, then concentrate on the burst mode test, doing it over and over and over, until you RESET or power off. The screen will "flicker", and by optical illusion appear to roll; the program inverts black to white to black on the ST screen while it's running to indicate it's *still running* (as opposed to crashed). (Besides, it looks neat.)

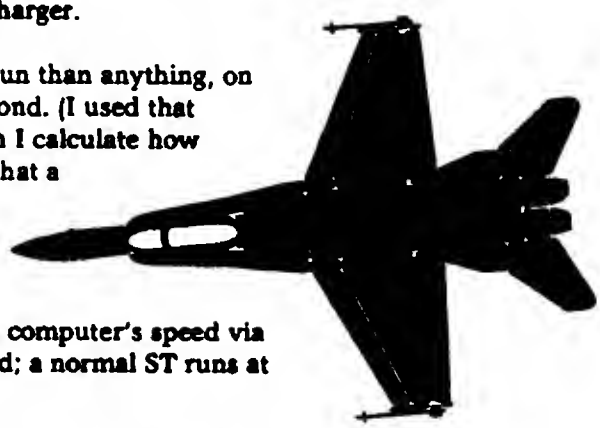
Other Editor: I admit it; I needed a way to show the program was working, that I could put in real quick; but I liked the effect so much, I never changed it!

Actually, you're seeing something amazing. Most Atari programs are well under 350,000 bytes long - that's one single-sided disk full. Yet you are seeing either a 4 or 8 million byte long program run, every single byte, in under a second! That's the power of the 68030 and burst mode, with SST RAM, unleashed all the way; that's the ~~gas-pedal~~ throttle all the way ~~down~~ open and the ~~supercharger~~ afterburner on max boost.

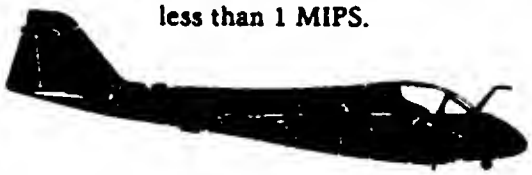
Other Editor: Oh, okay, you can have the airplane metaphor if I can have the next paragraph.

Editor: It's a deal!

Optional Activity: Watch first few minutes of "The Road Warrior" (with Mel Gibson). The cogged-belt driven thing on top of the engine is the supercharger.



As you'll see, during burst runs I keep statistics, more for fun than anything, on how many "ticks" each run took; each "tick" is 1/200th of a second. (I used that particular ticker because the ST conveniently provides it.) Then I calculate how many true 68000 instructions were executed, bearing in mind that a 68000 instruction is typically 2 bytes long, and print the results; it varies from 5 to 8 million real 68000 instructions per second, depending on speed, RAM, and whatnot. This value, Millions of Instructions Per Second, is called MIPS by computer types, and is one fairly standard way of advertising a computer's speed via a benchmark. An 8 MIPS machine is *very fast* indeed; a normal ST runs at less than 1 MIPS.



MIPS is useful inside this test as a measure of how the 68030 and memory are performing. If you're interested, write down the number of MIPS reported by the burst mode test, then cut the wait states by 1; you'll see the MIPS jump up dramatically. That's real machine speed improvement that you will see in *your* programs... provided the machine stays solid, of course!

Optional Exercise: Wiggle the mouse around while the test is running. You'll see some fun effects as the mouse clashes with the screen inverter in SST RAM. (You will also see MIPS drop, which tells you I'm being honest in the benchmark; the mouse interrupt is taking time away from the test.)

Now, if you want to find out if burst mode is solid, wait until everything gets nice and warm, and see if the program is still running. It's designed to crash if it gets a even a marginal access from RAM, and once again, these components are less tolerant of flakyness when they're good and hot. If it's still hammering away when warm, you're in good shape with 2 burst waits. If you're solid at 2, well, why not go for 1! (At 20 MHz you have a very good chance of it working at only 1 burst mode wait state.)

Editor: Oops... I guess I got a little carried away!

So go to 1 burst wait state. Or 0 burst wait states. Subtract burst mode wait states until you actually see a crash happen; usually the SST RAM will survive the low speed tests, since they're not much of a challenge, but will fail on the medium or

high speed tests, where they're running a program out of SST RAM. Those stress SST RAM the very most. I'd like you to get to where you see a crash, then back off 1 wait state, and do a long test (overnight is good) to make sure burst mode is solid.

Tuning Normal Wait States

Okay, you've found your burst mode limits. Now let's crank down the normal wait state limits. Leave your burst mode wait number alone, take your normal waits, begin at 2 (you've already tested 3 while optimizing the burst mode waits) and work down to a crash; add 1 and test. You will probably also want to remove the "D" character from the name as you're not as interested in burst mode tests as you are regular mode tests.

Okay, we've determined the normal and burst mode wait states, and only you know the oscillator value you're plugging in... Copy the RAM tester program somewhere, and rename it to: RAMnbmm.PRG (e.g., replace "TST" with "RAM", and get rid of the "D").

This becomes our standard SST RAM initializer. You run it any time you want to "switch on" SST RAM; until you run it, SST RAM is off.

Again, if you're going to use this as your standard RAM initializer, it must be in its folder without any other programs named RAM(anything).PRG or TST(anything).PRG.

At this point, you are likely to have a solid SST system that is working at its maximum performance. However, if you start running into odd crashes that don't repeat predictably (e.g., the same program crashes only once out of several times), you may be right on the edge of a needed wait state; try bumping up the *normal* (not burst mode) wait states by 1. I have seen this several times; there are interactions with the ST hardware that are very difficult to duplicate in the SST RAM Initializer/Tester, and perhaps something is out there that can use an additional "tick".

I have actually seen 0 burst wait states work, but the system didn't stay stable very long. I cheated; I ran fast SIMMs, low MHz, and still I had to keep hitting the 68030 and SIMMS with "freeze mist" (which cools things down way below freezing) to keep the SST running. This is not an economical way to compute. Also, it makes frost, literally water from the air condensing and freezing onto the board, which makes for short circuits.

If you're running around 20 MHz, I would recommend testing to see how far down you can go; you may well get by with almost no wait states of either kind.

Again, the oscillator value must be over 16 MHz for the SST board to operate. This is because the ST is running memory at 16 MHz and we must synchronize to it as needed (we have to grab the signals *first*); the synchronizing requires we be at least a tad faster than the ST. Consider 18 MHz the absolute minimum speed; it won't hurt a 16 MHz 68030 to run slightly faster, at 18, or even 20 MHz.

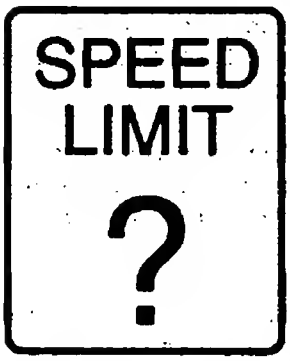
I have taken 16 MHz 68030s and run them at 33 MHz; they work until they warm up. You can "over-clock" a 68030 CPU because of the built in safety margins, but the chip will get *hot*, and hot chips fail. It usually doesn't kill the 68030 the first time (but those chips are expensive); if you want to change the MHz, all you have to do is *change the oscillator* (which is already socketed!). In a later chapter is a list of recommended CPU and corresponding oscillator speeds/speed limits, and part numbers.

In everyday work, I run a 33 MHz 68030 at 33.3333 MHz, and use 3 & 2 wait states without problem on 60 ns (nanosecond) SIMMS. This is just to give you a starting point. 3 & 2 is maybe conservative; I got that way testing the first prototype SST boards, which were not as solid on timing as the final product is.

Up at 40 MHz, our testers report that 60 ns SIMMS really help. With 80 ns SIMMS, they run 3 & 3 wait states; with 60 ns SIMMs, they run 3 & 2 wait states, and some of

them run 2 & 1, and the machine really flies. The burst mode performance test reports around 7 to 8 MIPS in that setup.

Editor: I guarantee "Your Mileage Will Vary", so you really should take the time to tune your SST, and see where it tops out.



Top Speed: Unknown

Since we have not gotten oscillators above 40 MHz yet, we don't know the top end of the SST, really. ***It positively will vary from board to board.*** The board's design limit is 33 MHz because of the 33 MHz DRAM controller; however, the programmed logic chips on some boards will probably take a few more MHz before going unstable, because we made a point of using slightly over-spec parts (for reliability.) Our feeling here is that 40 MHz is probably a "realistic" limit, but again, no one knows. Oscillators are around $4.00 here, in Denver, the backwater of the computer world, and cheaper in surplus shops; grab a few different values and try them out! There are addresses for places to get oscillators in a later chapter, if you can't find any locally.

We do not believe that most boards will take 50 MHz, if you're wondering; that's stretching the speed-of-light limit and our chips *way* further than we ever expected to go. We might have to change the board layout substantially to prevent power supply ringing, go to even faster glue logic (which has its own problems), faster drivers (which start ringing), and other madness. In other words, we would have to increase the cost of the board substantially, to the point where we don't think it would be worth it to you.

As I mentioned before, the memory controller is rated at 33 MHz, and we do not know how much it can be "pushed" before it fails. We do know that it is comfortable at 36 or 38 MHz and some of them will run 40, so anything is possible. A 40 MHz version of the memory controller has *just* been released, so, who knows? (I'm sure there'll be more information in the next Gadgets News-Herald-Update-Enquirer.) This is so new, it barely made the manual!

But MHz is not everything; the CPU to memory connection is where you get your performance. That's called "bus bandwidth" and is the basis for speed in computers; the faster you can move memory around, the better things are. The 68030 and the SST RAM are pretty good at it. If we keep turning up the 68030's speed, sooner or later the efficiency of SST RAM falls off (you have to add more wait states), and the overall horsepower of the machine drops. A *properly tuned* 33 MHz SST will *outrun* an out of tune 40 MHz SST, period, in any decent benchmark or real-life program around!

Or, if you drive cars, you have probably noticed that on most engines, as you approach redline, your power really falls off, and it's time to shift; that's because at that speed, the engine can't get enough gasoline and air into it, nor exhaust out of it, to be efficient and make power. The 68030 and its SST RAM memory system are like that, too.

I'll repeat the above caution once more: The testing program gives a *good* overall picture of how fast you can push things, but I have seen it sometimes be a hair optimistic on what you can get away with in wait states. If you see annoying, persistent crashes, particularly of the Flat Line variety (11 bombs), add 1 to your normal wait states. If the problem persists, add 1 to your burst mode waits.

Anytime you change MHz, add or change SST memory, please re-run this testing sequence. Yes, do it going from 4 to 8 megabytes; you are loading your power supply more. And different SIMMs bearing the same speed rating are practically guaranteed to have different true speeds. I also make it a practice *never* to mix chips from a different manufacturer; I'm not *sure* it matters, but...

SST Flexibility

We don't force the SST RAM on during power on, like we could have, because we believe in giving you some flexibility. If SST RAM is active by the time the "Desktop" shows up onscreen (which means, if you put the RAM initializer into the AUTO folder), many things are done differently. For instance, desk accessories are loaded into SST RAM, and SST RAM is commonly used as a temporary memory area. This may or may not be what you want (particularly if a desk accessory doesn't work in SST RAM!) So we give you the option and let you configure what works the best for you.

Again, if you have the RAMnbmm.PRG file in your AUTO folder, you must *also* have a copy of it on your "C:" directory, so that it can "find itself". Also, make sure that if your disk driver software needs the DISKxx program (Atari HDX 4.03 and 5 *do not need* it), that DISKxx is in the AUTO folder too.

Since you need to run RAMnbmm, and *then* DISKxx in that order, you will need to remove everything from your AUTO folder, and copy in RAMnbmm, then DISKxx. This will make sure that they are run in the correct order. Next, you can copy the rest of the stuff you want into your AUTO folder.

If you have RAMnbmm in your AUTO folder, and want to startup *without* initializing SST RAM, press either "SHIFT" key during bootup. This will disable the SST RAM initializer, and you will only have ST RAM when you start up. Remember, you can run RAMnbmm anytime, just by double-clicking on it.

Eric Does Nikola

Last year, my son Eric, who just turned 10, had a school project called "The Famous Person's Tea". This is a yearly event in which each student selects a Famous Person from history, dresses as that person, and gives a 10 minute memorized on-stage speech about that person to an audience of around 200 parents and relatives.

I know adults who would rather do anything than give a speech to 200 people! Dale Carnegie did research and found that fear of public speaking is one of the greatest fears of people, exceeding things like divorce or slow death by torture. And it is so very hard, as a parent, to watch your child out there on stage, maybe ready to freeze up or forget the speech or in general do what adults have been known to do in this situation. (I help teach public speaking; I have seen grown adults break into tears and run out of the room while giving a speech to a mere 30 people!)

I am not that sure I agree with the school giving this as an assignment; not all the kids made it through their presentation, for instance, and that is a hard thing to overcome later.

So my son came to me and said he was having trouble picking a Famous Person – "Dad, who do you think is the most famous person?"

I suppose other people might say Leonardo da Vinci or Einstein or FDR or Oliver North or Nelson Mandela or Boris Yeltsin or Madonna (it takes all kinds, okay?). But without an instant's hesitation, I said, "Nikola Tesla". Here is why.

Nikola Tesla

99 % of the people in the world have never heard of Tesla. The remaining 1% hold him in the HIGHEST of respect. Let me tell you why.

Tesla did his work in the late 1800's. Among other things, Tesla invented and helped market, over Tom Edison's strong objections, Alternating Current – A.C. current, as it is called. Edison wanted D.C. current. Because Tesla won, you have the electricity you're used to, at your house and office, and you have computers. A.C. can be sent thousands of miles, stepped up and down in voltage, sent very efficiently (via Tesla's invention, 3-phase power.) Tesla holds the invention / patents for things like A.C. generators, transformers, the A.C. motor (think of every motor in your house), and so forth.

Did I mention Tesla invented radio? Most people think Marconi did this; in 1943, the Supreme Court ruled Tesla's patent overrode Marconi's. By this time Tesla was gone, which will tell you all I think about the pace of the law. In the early 1900's he demonstrated radio-controlled ships to the Navy, for instance.

Tesla was from Serbo-Croatia, thence called Yugoslavia, and now, most sadly, in the midst of a civil war to split back to Serbo-Croatia. (There is a highly regarded Nikola Tesla Museum in Belgrade; it is among my dearest wishes to visit it!! He is a national hero there, justly.) Tesla came to the United States to work for Edison with this recommendation from the the Paris office of General Electric:

"Dear Mr. Edison:
There are two real geniuses in the world. You are one. The young man who gave you this letter is the other."

Tesla did a job for Edison and was offered a $25,000 bonus if he could first do it (it was regarded as impossible) and second, do it fast. It was soon complete, and Edison, who had made

sure not to put the promise in writing, didn't pay; he simply said, "You don't understand our American humour, Tesla."

Very funny. Tesla quit on the spot, an event that cost Edison billions in the long run, and spent time literally ditch-digging. But it worked out; the foreman of the ditch-digging crew knew George Westinghouse, another electrical pioneer.

In the late 1800's, Tesla teamed with George Westinghouse, who founded Westinghouse™, to bitterly oppose Tom Edison's General Electric™, that was pushing D.C. power. D.C. is the worst; for instance, you'd need to live within 50 miles of a power plant to even have electricity, and people in rural areas could not have power. A.C. can go thousands of miles with no trouble. Edison was much more well known than Tesla (which still remains true today). And "The War Of The Currents" was played out in the 1880's to determine if mankind would have the limited Edison D.C. or the no-limits Tesla A.C.

It was played dirty, like many politics. Through a political crony Edison arranged for the first electric chair, at Sing-Sing prison, to use A.C., then trumpeted about how "deadly" A.C. was; "After all, they use it at Sing-Sing." (In truth D.C. would work equally well). "If you install A.C. in your house, within a month someone will die of it.", Edison said. You and I use it all the time today. (In reality, D.C. and low-frequency A.C. are equally deadly; don't get either one into your body.) High-frequency A.C. is pretty harmless.

Nikola Tesla was furious at this sort of dishonesty – and had a little personal score to settle, too. So at a World's Fair Exposition, he won a contract to do all the lighting (a new thing!) with A.C., and at hundreds of demonstrations, Tesla passed A.C. current through his body, and with it lit hundreds of light bulbs, melted wires, spun motors, and many other things. Thus he proved it was "not deadly".

The secret? Tesla did this by using high frequency A.C., instead of low frequency, the 50/60 Hz used now; for heaven's sakes, don't plug yourself into your house power! (But see below on the Washington show.) And while Edison was a home-grown folk hero and Tesla was some unknown immigrant, gradually, Tesla's style won people over.

But it was rough on Tesla and Westinghouse. Tesla had licensed his many patents to Westinghouse for a dollar per horsepower; Westinghouse ran desperately low on money during a particularly bad Edison assault. Westinghouse, in despair, came and talked to Tesla, hoping to renegogiate to save Westinghouse Electric. Tesla gravely listened to the money problems, got out his license agreement with Westingouse from the safe, and said something like, "Mr. Westinghouse, when no one else would listen to me, you believed in me. Give my A.C. power to the world."

And he tore up the license agreement, giving Westinghouse A.C. power for free.

Because of this, Westinghouse, A.C. power, and the electricity you can so handily use today survived. Because of this, Tesla later ran out of money and faded into obscurity and even poverty. (Edison? General Electric gave it up, switched to A.C. power, and is still around; Edison is regarded as a hero. They make movies about Edison... sigh... although there is one now about Tesla (not real good, in all honesty; I hoped for more)).

Those patents are beyond estimation in value today. Hundreds of billions of dollars of A.C. power, and the derivatives of it, exist; it is the world's power system. So in a vital way, Tesla WON, but at a terrible sacrifice.

Yes, Tesla could have negotiated with someone like Edison for the A.C. patents; instead, he had honor.

There is no question in my mind that **Nikola Tesla is among the most VITALLY IMPORTANT, unknown heroes in history.**

I will guarentee you a fascinating time if you read the books about Tesla in a library. I am at best giving you the slightest bit about Tesla in this interlude; there is much, much more.

Tesla's work is still of extreme significance to us today. The "Tesla Coil", a high frequency, high voltage resonant transformer, which I demonstrated at the Washington Atari show two years ago, makes six-inch sparks I can draw to my finger without feeling any shock; it's high-frequency A.C. and never penetrates the skin. In fact, we lit up neon lights through a chain of four people holding hands, and lit up flourescent light bulbs by just holding them in our hands, no connection needed. I have never run a more crowded, popular show booth!

Nearly every CRT (including the ones for the

ST) uses a Tesla Coil variant to generate high voltage for the picture tube. Tesla's work provides the power for your life in a variety of ways, from power generators ("alternators" in your car) to transformers to transmission systems to the motors in your house to other things I have no room to write about. You don't know it, but Tesla has a heck of a lot more influence on your life than Edison does. Tesla changed the world.

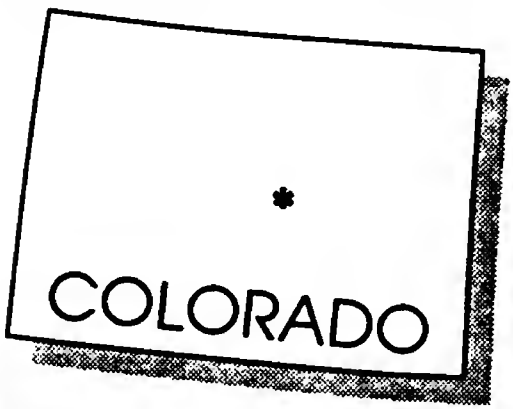
And as I've written about in Current Notes, a fine USA Atari ST magazine, Tesla's work can be implemented today in a variety of useful ways. For instance, Tesla set up an electrical oscillation involving the entire planet in 1899 (the ground is extremely conductive, if you didn't know, especially the molten-iron core of the planet). This is the only known time this has been done. It was part of an experiment to provide free power to the world; in Tesla's dream, drawing power would just require hammering a spike into the ground anywhere, connecting to it, and turning on what you wanted to use. No wires back to some generator somewhere; the generator uses the ground for its wire! It could still be done and there are still people trying. Tesla's work also provides a viable method of quickly regenerating the ozone layer, which chlorine molecules (CFC's) are busy nibbling away at; Tesla even wrote specific directions on how to do it.

I know these last two sound pretty much like science fiction. But given as how what Tesla did WORKS, and I'm writing this on a computer depending on many Tesla inventions, I believe. Tesla was in the process of wiring up the power system, at Wardenclyffe, when he ran out of money; it was all designed.

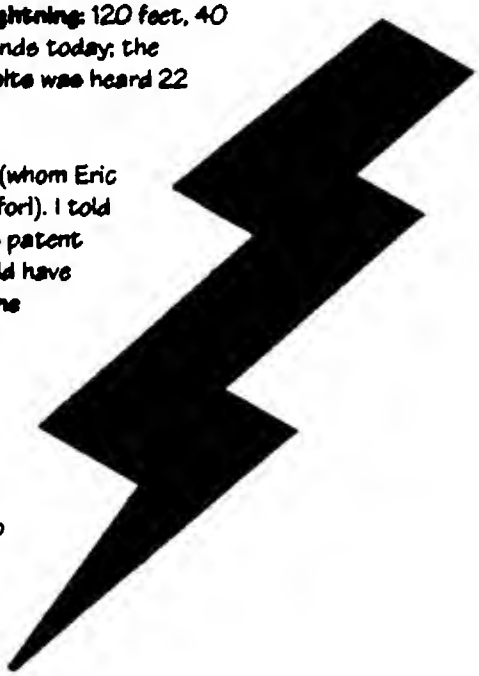
Back to Eric

Eric had never even heard of Tesla, like most folks.

So I drove my son just 50 miles south, to Colorado Springs, Colorado, where Tesla had his laboratory nearly a hundred years ago. I showed him the beautiful memorial sign erected near the spot by the Tesla Society.



And I told him the magnificent story of the night in summer 1899 when Tesla set the world's record for man-generated-lightning: 120 feet, 40 meters. (The record still stands today; the thunder from his lightning bolts was heard 22 miles away, in Cripple Creek).



I told him about Edison (whom Eric still has grinding contempt for!). I told him about Tesla giving up his patent royalty money so people could have electricity, which is among the major reasons why life has improved so much for everyone since Tesla began inventing... over 100 years ago.

By the end of our trip to Colorado Springs, as I pointed out where it had all happened, Eric's eyes shone with fire, and I knew in my heart that we had another soul in the family determined to push the envelope. Wait another 10 years, folks.

And he said, "Dad, I want to be Nikola Tesla at the Famous Person's Tea."

The Presentation

Any parent whose child has been onstage knows the heart-in-throat feeling. Eric, as a "T" ("Tesla"), was late in the presentation by alphabetical order. Every parent was brought together, all pulling for the kids to make it through their presentation, alone onstage, with a microphone. Most kids did well. But there were others; kids before Eric forgot their speech, one broke into tears!

Eric came onstage; there was a roaring in my ears of pure parental stress. He was dressed as Tesla dressed, in a rented tuxedo, complete with gloves, cane, and a tophat, and I got, for one moment, a warm glimpse of what my ten year old will be as a man. A voice behind me, someone I don't know, said quietly, "My God, he is handsome."

He stood there a moment, looked right out the crowd, and hesitated; my heart froze.

Eric Does Nikola

And without a quaver in his voice, he began. He'd hesitated on purpose, to get the crowd's attention!

"My name is Nikola Tesla.

I gave electricity to the world..."

and so forth into his speech. He did it perfectly.

The audience listened. Sure, I'm a proud parent, but they were hushed and listened with honest interest to this amazing and true story. And at the end, they applauded and some cheered...

And as I write this, it's like it was during his talk; tears rolled down my cheeks, then and now, remembering.

I don't know if I will ever be so proud of anything ever again, but the moment will sing in me forever.

Take Off

Be certain your restraint harness is securely fastened. Push the twin throttles forward to the Afterburner detent, and then all the way to Full Throttle. The SR-71A will rapidly accelerate. Caution: Your acceleration and pressure suits should be fully operational at this time, lest you lose consciousness during takeoff.

Also, do not attempt to exceed Mach 3 at less than 40,000 feet. People have been complaining about the "sonic boom".

Flying with Your SST Day to Day

Now that you've got your SST installed and tuned up for performance, it's time to learn how to use it from day to day.

The SST could be thought of as adding a much longer gas pedal to your ST. If you press the pedal the same old way, the SST will run a little faster than ST speed. (At that point, you'll sure be wondering why you even bothered with installing it!) You now need to learn how to push the gas pedal deeper, to kick on the SST's speed and *afterburners*.

Editor: I won! I won! Dave used an airplane metaphor!

The Cache

You'll notice after you've started up that there are a great many more entries on the desktop menus than there were on the older TOS. This is the TOS 2.06 desktop; this new desktop can do all sorts of neat things, like search for a file by its name. It has all the editable icons and whatnot of the TT desktop. I'd tell you all about it - except, in all truth, I don't know as much as I should about it! I'll have to refer you to Atari's manuals on the desktop, at least until I learn more about it. Most of it is very intuitive and easy to use.

Gary Heidlebaugh, suited up

We want to look at the CACHE entry for a moment. The "CACHE" on/off menu entry is where the "BLITTER" entry used to be (the OPTIONs menu), fourth menu from the left, last entry.

If you run without the CACHE on, you will run at essentially ST speed (because of video contention). If you run with CACHE on, you'll run at about 4 times ST speed, with some things slower, some faster, depending on what the program you're running does.

The CACHE entry also controls "burst mode status". If the desktop shows the "CACHE" on, and you've initialized SST RAM, you can run in Burst Mode (and gain even more speed, real warp-drive stuff); if the CACHE entry is off, burst mode is also off. If you're running in SST RAM without burst mode enabled (e.g., without CACHE entry on), you're missing out on the maximum speed of the SST.

Just to remind you again, the Shift – Control – Alt – Delete keypress does not work to reset the SST. Check the READ.ME file for the latest information.

Technical Note: Selecting the CACHE entry enables the Instruction Cache, Data Cache, and Burst Mode.

Saving The CACHE Status

The status of the CACHE entry is saved along with the DESKTOP information whenever you do a "SAVE DESKTOP", along with whatever icons you've got on the desktop, the positions of the currently opened windows, and whatnot. In older TOS (TOS 1.0 - 1.6, found on the original 520 ST through the "non-Mega ST" machines), this is saved in a file called DESKTOP.INF. In those Atari's with the newer TOS (TOS 2.0 through TOS 3.05, which are SST, Mega ST, and TT), the DESKTOP information is saved in a file called NEWDESK.INF.

So, if you turn the CACHE entry on or off, then do a SAVE DESKTOP, the next time you power on, you'll get the CACHE the same way you had it set last time. This means you don't have to go to the CACHE entry to turn it on *each* time you start up; it turns on automatically.

There are two places this NEWDESK.INF file can be saved; it depends on whether or not the SST system thinks you're running with or without a hard disk. The NEWDESK.INF file is saved on floppy A: if you are only running a floppy system (no hard disk), or saved on hard drive C: if you're running with a hard disk.

The NEWDESK.INF file can get damaged, in my experience; if this happens, strange stuff will happen. Typically, you'll "lose icons", and you may not be able to start up at all without resorting to using hard disk utilities. This is where a backup copy of NEWDESK.INF can be really helpful! Just copy the file (to, say, NEWDESK.BKP). Then, if NEWDESK.INF dies, delete it, and copy a new one from the backup.

SST fastRAM and ST RAM

The first thing to remember is that the RAM in the SST is not just a simple memory upgrade. It is a completely different kind of memory, much faster than ST memory, and has some restrictions, in order to maintain that speed. For instance, SST memory cannot be used to display video; if it did, it would be slowed down, like ST RAM is. And SST memory also cannot directly access the floppy or hard disk.

Don't panic; this doesn't mean you can't load a program into SST RAM from disk; it just means we had to be sneaky about it. We handle this case pretty much by using ST RAM as an intermediary; for instance, to read from disk into SST RAM, we read first to ST RAM, then copy to SST RAM with a special fast copy routine. SST RAM flies and is really fast.

Hence your Mega 12 (assuming a Mega 4 and 8 megs of SST RAM) is really a Mega 4 and a SST 8!

Editor: I knew it. He wouldn't be able to resist.

Now here's where I'm going to have to bore you with some technical stuff. Programs *assume* that they have a continuous memory space to work with. This means *no gaps*. They often can't even handle the concept of two memory spaces. Surprise! There is a gap 12 megabytes long between ST RAM's end and SST RAM's beginning; this is the same way the TT architecture is set up.

The way TOS 2.06 "resolves" this is to let you select either SST RAM or ST RAM for your program to load into. Well, this is fine; if your program runs in SST RAM, it'll fly fast indeed. (If it runs in ST RAM, with CACHE on, it'll fly medium fast.)

Right at the moment, there *isn't* a way to make memory "continuous", so that your programs "see" 12 megabytes of contiguous RAM, with the ST and SST RAM all tucked together.

Other Editor: There may be a way to do this; I'm working on one, but this is mighty voodoo stuff, and not working yet. The ST was never designed to do this, but that's never stopped me before. Keep an eye on fithe READ.ME file on the release disk, okay?

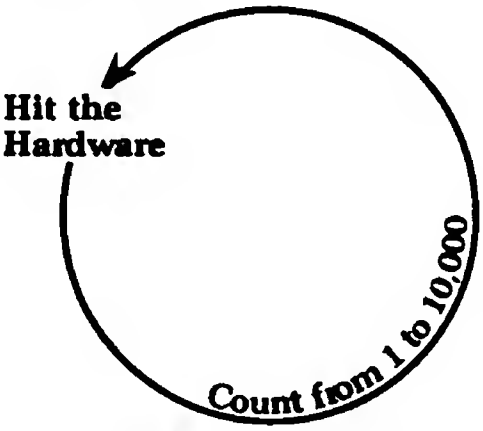
Anyway, you have two kinds of memory, ST and SST, as well as the cache (which is internal to the 68030). Whether or not your programs are going to run is going to depend on how well they're written; if they contain special types of bugs, they'll crash in various situations with the SST. So anyway, programs are like driving: "The Slower You Go, The Less Likely It Is You'll Crash".

Why is this, anyway?

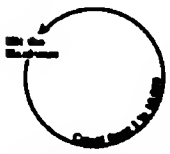
Some programs use "timing loops" to cause speed delays. For instance, many of the hardware chips of the ST can only be accessed every so often, and must have delays between accesses. So, some programmers do a delay this way:

1. Hit the hardware
2. Count from 1 to 10,000 (this takes, say, 1 second on a regular ST)
3. Hit the hardware again...

and so forth. (Illumination 24.)



Illumination 24 - An ST Timing Loop



Illumination 25 - The Same Loop on the SST

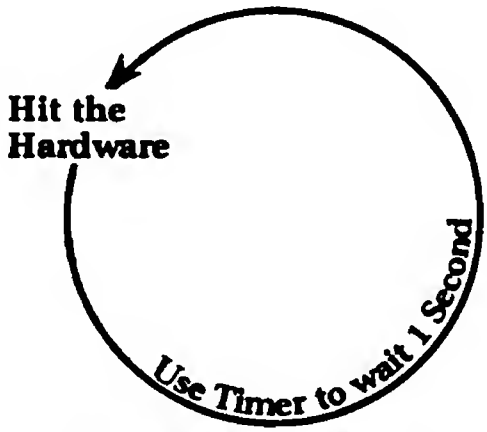
On the ST, this works fine, and on a throttled-back SST, this is also *likely* to work fine (running in ST RAM with CACHE off). However, when you put this program in SST RAM and turn on burst mode, that timing loop won't take 1 second; it'll take 1/10th or so of a second to run. (Illumination 25.) The ST hardware will get hit too fast, and the hardware, and therefore the program, will fail.

Editor: Bombs. Freezes. Doesn't work.

Another real bogeyman is disk copy protection, particularly on games. Many of these schemes rely on special marks on the disk, which are written with particular emphasis on timing (to make them tough to copy). Again, the timing loops will foul up at higher speeds. More intelligent software (and anything accelerator or TT compatible) does this:

1. Hit the hardware.
2. Use an outside "Timer" to wait 1 second.
3. Hit the hardware again.

The outside timer, typically the MFP 68901 chip, counts 1 second regardless of whether it's running on the SST or the ST, so everything keeps working; this program would work just fine at 150 MHz! (well... until the 68030 melted down, at least!) (Illumination 26.)



Illumination 26 - An Intelligent Loop

This is what's been done to most ST software to upgrade it; which makes it compatible

with 68000 accelerators, and also the TT.

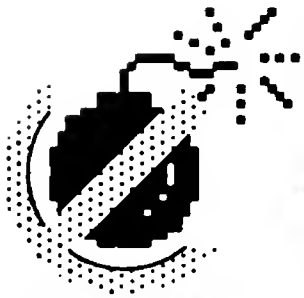
Because there've been accelerators out for awhile, most programs are "aware" of them and use intelligent timers, not unintelligent (programmer talk: brain-dead) delay loops. Hence most software runs just fine at high speed, in SST RAM, at full acceleration.

Let's talk about the different modes your SST machine can be run in and what programs are likely to work in each.

ST RAM, No Cache (Low Speed)

Let's begin with non-CACHE mode and ST RAM, which is the slowest of all modes the SST can be run in. It performs right about the same as an Atari. We are essentially choking off the 68030's instruction input down to around 8 MHz. To get the SST to go this slow, you have to manually shut off the CACHE via the OPTIONS menu.

Because some programs were written with the assumption that the ST would never get faster (which the programmers were warned not to do, but did anyway), they can only run in this mode. There aren't very many, fortunately.



The Ultimate Goal...

Lots of programs will run in this mode, where you're in ST RAM, with the 68030 not being helped along by its internal caches. Fortunately, most of these programs will run in a faster mode, too!

The exceptions are programs that rely on running with a 68000, *not* 68030, chip and do things they should not; Atari's Developer Notes have long advised developers not to assume that the CPU chip would be a 68000. If a program is going to die, you're likely to see 2, 3, 4, and 11 "bomb" errors; this means the program is written in a way the 68030 doesn't like.

This isn't the end of the story. I'm working on compatibility software that helps such programs along by pampering them and making them believe they're in a 68000 environment. This has several pieces and is pretty complex, and (*sigh*) will not be finished for the first release of the SST software – but it's on the way.

ST RAM, with Cache (Medium Speed)

Now, let's move to CACHE ON mode and ST RAM, which is the sort of medium mode. You'll run around 4-5 times as fast as a regular ST in this mode. And all we're doing here is turning on the caches internal to the 68030.

Again, lots of programs will run in this mode, no problem. The exceptions are programs that won't run on a 68030 period, and programs that rely on slowed timing (see above), which in general would crash on any faster ST system. (With the advent of accelerator boards for the ST, most of these programs have been fixed, but *not* all of them.)

One other exception is programs that modify themselves on-the-fly, which upsets any cache mechanism.

SST fastRAM, Cache On, Burst Mode (High Speed)

In the first two modes, we were running with a program located in ST memory. As I've talked about previously, any such programs are limited to 8 MHz speed due to video contention, even if you've got a 40 MHz 68030 in there!

In this mode, we tell the SST that we'd like the next program we run (double-click on) to run in SST RAM, not in ST RAM. We also have CACHE turned on from the OPTIONS menu, since it not only turns on the caches, but also enables you to do burst mode.

You will notice some *very* snappy performance in this mode! This is because ST RAM

isn't slowing down the program anymore. No more video contention! This is what the SST is all about.

Most programs have run okay in SST RAM, in our testing. The exceptions are programs that are doing something wrong; if a programmer "followed the rules" of ST software development, their program should work just fine in SST RAM. Even if they've "bent the rules" we try to step in and help the program to run. Later on, we'll get into some down and dirty technodetails to help you understand about those "incompatible" programs.

Compatibility: ST RAM, Cache, SST fastRAM

What you have to find out, for *each* of the programs you want to use, is if the program will run in SST RAM (best possibility), or in ST RAM with cache (still pretty good), in ST RAM with no cache (same as ST), or just plain not at all.

If you should run into a game or something that breaks when you try to run it at high speed, try running it at low speed; in other words, turn off the cache and run it in ST RAM. This will get you pretty close to ST speed. It'll likely work, as all of its "delay loops" will be running at pretty much the same old speed as before.

Finally, as I've no doubt mentioned before, there are going to be some programs that *cannot* work, for various reasons. Some fail because they *are not compatible* with the 68030. We make an effort to fix such programs "on the fly", but sometimes the programmer interferes with even that, and the program fails. There are also programs that only run on one version of TOS (say, 1.2) and which will fail because the SST is running TOS 2.06. This seems to happen the most with games.

In our experience testing the SST across many different machines, we've found most programs work in SST RAM, but there always seem to be a few that sputter and die.

Editor: Stall and crash?

So, some programs work at high speed (SST fast RAM plus CACHE on) or medium speed (only CACHE on); but some don't, and need lower speed to operate (no fastRAM or no CACHE). Hence, we need a way to *automatically* tell the ST, for a given program, to use ST or SST memory, in order to tailor the speed for the particular program.

You already know how to turn the cache on automatically; just pull down the CACHE entry and select it, so it has a checkmark by it, and save the desktop. Telling programs they're supposed to run in SST RAM is a bit more tricky; this is done by changing an SST RAM Yes/No "switch" inside each program. Computer types call this "setting a flag", which indicates that the program should either be loaded into SST RAM, or into ST RAM. The preferred program to do this is called PRGFLAGS.

PRGFLAGS

You tell a program to load into SST RAM by "setting a flag" that is stored *inside* your program. There are actually three such flags, which can be changed with the program PRGFLAGS (written by Atari). Each "flag" tells the ST what to do when you run that individual program, and are stored as "bits" inside each individual program.

Hence, whether or not a program will load into SST RAM, or whatever, isn't instantly visible by a look at its icon or something easy. (I wish it was; and maybe I'll do something about that someday!) Instead, you have to use the PRGFLAGS program to look at or change these flags.

Clear Memory Flag

The first bit controls whether or not RAM under a program is cleared before the

program is loaded. Some programs need this. Most don't. It can save a lot of time in loading. You've seen this before in the PINHEAD utility from Codehead, and it works well.

I personally don't use this simply because I don't want to have to wonder where a crash came from; I crash enough as is. Every now and then a program relies on memory to be zeroed out as it starts, and can randomly crash if this is not done. Such crashes are *very* difficult to pin down - especially if you're not sure if the *hardware* is at fault!

Default is to zero out RAM. Change it with caution!

ST or SST RAM Flag

The second bit controls whether or not a program loads into SST RAM or into ST RAM. Well, this is, of course, provided that there's enough SST RAM available to load into! You can check available ST and SST RAM with the CPX (Control Panel Extended - select "General", then "Status"), or with MEMSTAT.PRG (on the disk that came with your SST). The default setting sets the RAM choice as ST RAM.

If you have a program that definitely hates to be run from SST RAM, then you can force it to only load into ST RAM with this flag. And you'll use this flag to test different programs to see if they'll run in SST RAM (which you're probably going to want to do).

Remember, SST RAM is where the speed is; the cache helps a little, but not nearly as much as burst mode from SST RAM.

ST or SST RAM Memory Request Flag

The third bit controls whether or not the program gets SST RAM or ST RAM if and when the program does a memory request (request a block of memory for the program). This is *not* the same as loading into SST or ST RAM!

Programs often request memory to set up tables, hold temporary areas for word processing, spreadsheets, or to setup an alternative video screen area. Ordinarily, if you're running a program from SST RAM, you'd want a program to get SST RAM when it requests memory, since SST RAM is so much faster than ST RAM.

Such a memory request is called Malloc, for Memory Allocation, a typical programmerism.

However, this setting *can* be used to make a program that isn't 100% SST RAM compatible still work. For instance, many programs do a memory request to get room for a video screen, which would fail if it was allocated from SST RAM (SST fastRAM *cannot* display video or do direct disk access, to ensure it is high speed availability for the 68030.)

This option lets you load the program into SST RAM, and run it, but *any memory the program requests while running will come from ST RAM.* Sometimes, this can make a program work from SST RAM.

The default setting is ST RAM.

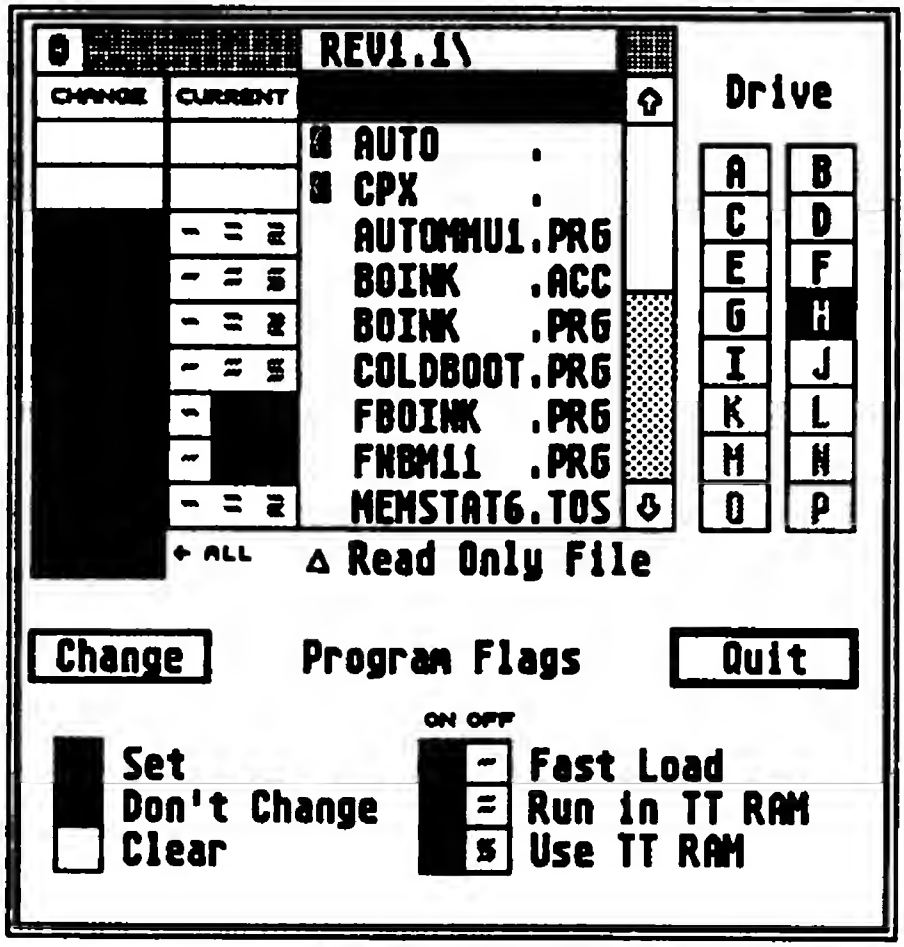
Incidentally, this PRGFLAGS scheme is the same way programs on the TT are selected to run in TT RAM. TOS 2.05 and 2.06 are really a derivative of TOS 3.01, for the TT. I'd expect to see a list of SST RAM compatible and incompatible programs available for users... someday. We're trying to build up such a list right now.

Changing the PRGFLAGS Settings

These flags (bits) are stored inside *each individual program.* If you copy a given program around, you'll copy its program flags with it; they are part of the program! However, only

one of the programs will be affected by PRGFLAGS. For example, if you flag one copy of a given word processor to load into SST RAM, any other copies are not automatically changed. If you copy the word processor *after* you have set the flag, the new copy will have the same flag settings as the old; it will *not* have the default settings!

Fortunately, PRGFLAGS is really pretty easy to figure out, and works well with the SST. So, run PRGFLAGS.PRG. It will take a moment to read in all the directories on your hard disk (so a really *big* hard disk will take some time!). Then it will present you with a display very much like the one in Illumination 27.



Illumination 27 - PRGFLAGS Display

Go ahead and play with PRGFLAGS to get used to it. To switch directories, it uses a pull-down menu, very much like a Mac. REV1.1\ is the current directory, located on drive H; just click on it (and hold the mouse button down) to choose a different directory.

As usual with an Atari File Selector, all the folders are shown at the top of the scrolling list; the rest of the files are in alphabetical order. Scroll downwards with the mouse or scroll bar to look through all the programs on *this* directory level. Click on a folder to see programs inside it, and note that the folder name just went up to the top to show you it's the current directory. At the upper right of the PRGFLAGS display, there is a "DRIVE letter"; click here to change the drive you'd like to work with; if a drive letter is greyed out, it means you don't have it installed on your system.

At the left of the program names are their associated flags. From left to right, the flags are:

"–" : Fast Load (if it is set, there is *no memory clear* during load)
"=" : SST Load (if it is set, *load into SST RAM*, not ST RAM)
"≡" : Memory Requests (if it is set, *memory requests come from SST RAM*, not ST RAM)

If the background of the "dash" entry is black, that flag is "set"; if it's white, the flag is not set. The "CURRENT" column shows what the flags are set at for each program, right now. To change a flag, you click the selected flag in the "CHANGE" column (that's why they start out all grey: "Don't Change"); the background will change to black (set). Click again to change the background to white (unset), and again to go to grey (no change).

You don't *have* to change just one program's flag at a time; if you like, you can modify *more than one* of the flags in *more than one program*, as long as it's in the current directory. Use the "change flag" area to set or clear a flag, then press "CHANGE" to have PRGFLAGS go update the flags in your program(s). Until you push the "CHANGE" button, no flags are updated, so if you hit the wrong flag, just rotate it back to grey, and don't sweat it.

What Definitely Won't Work In SST RAM

There are some things that will absolutely *not work* in SST fastRAM. You need to know about them so you can keep an eye open for them, and make sure they run from ST RAM. As a general rule, they are programs that bypass the operating system (TOS) and talk directly to the hardware to do something.

If the program goes to the operating system, the SST works with it to keep things running. It is much harder, but still possible, to handle programs that go directly to hardware; I have not gotten that software up and running yet, but it *is* on the way.

Editor: Somehow, this sounds a lot like what Spectre does, when Mac programs try to access the ROMs directly. I guess that means "it can be done".

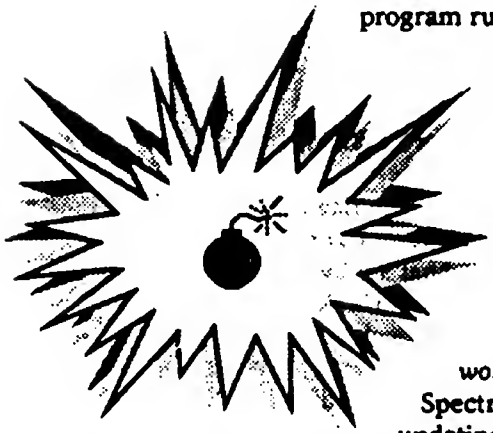
Programs are *supposed* to nicely ask TOS (The Operating System; yup, that's TOS) to do things to the hardware for them. We work with TOS, in order to make these programs function on the SST. For instance, if a program tries to read from disk directly into SST RAM using TOS routines, we tell TOS to safely read from disk into ST RAM. Then we fast-copy the data to SST RAM, where the program expects to find it. Having no "direct connection" between SST RAM and the disk doesn't matter, because the program finds the data where it's supposed to be.

However, if a program running in SST RAM goes directly to the hardware (and it can), and tries to transfer directly, that data will go to SST RAM, where it fails. If so, that program needs to be run from ST RAM, so the direct transfer also goes into ST RAM.

So, programs that directly manipulate the hardware to setup video displays will typically fail if run from SST RAM, because SST RAM doesn't allow video in it (which would slow it down). Typically, you'll see a really wild display, as the hardware shows you some random part of ST memory as a video display. Some of these are pretty spectacular. Invite your friends over to see it, if it's in color!

If you get wierd video when running a program from SST RAM, an alternative is to use PRGFLAGS to set the third "Malloc" flag so that when a program requests memory, it gets ST memory. Usually, programs that do this request memory for the screen area, and if we can force that request into ST memory, it'll fix the problem.

Second, programs that *directly* manipulate the hardware to read or write the floppy or hard disk *will* fail. This is because the DMA chip (Direct Memory Access, for disks) cannot work with SST RAM. If a program reads in a file or somesuch normally, no problem; we work with the operating system to route disk requests through ST RAM. But if a program running in SST RAM starts messing with the ST disk hardware by itself, it's in *big* trouble.



Such things as disk formatters, direct sector editors, track readers, "disk copier" programs and whatnot belong in ST RAM, period. When you hear the word "disk", get suspicious. If it seems like the program does disk "normally", through the 'ol File Selector or File Name (like a word processor or text editor), not to worry; if it's a low-level disk PEEKer, beware.

Third, programs that absolutely, positively require 68000 timing *will not work* with the SST, which is absolutely, positively a 68030. For example, Spectrum 512™ from Antic Publishing does this; the 68000 spends its time updating color registers in a tightly fixed, nailed-down-tight timing kernel. On the SST, this will break, because the 68030 will time differently. (Spectrum 512 will also fail on any other accelerator running at higher than standard ST speed.) It is likely other "many colors on the screen at once" programs will fail, too.

It is possible that these programs could be patched to work. In fact, with a 68030 there is more horsepower available, to make a program even more powerful! For example, VIDI-ST™, a hardware video digitizer, will fail because it also relies on exacting 68000 speed. This can be fixed only if the designers give me a peek into its workings... or, if they modify it to work themselves.

There are certain programs that simply will not work with a 68030 processor, period. (I have a fix in the works for them, but it's still "in the works"; it's very similar to Spectre's ZeroStore handler, and requires a lot of careful tuning.) If you run a program and get immediate bombs after it loads, you've probably found one of those "absolutely will not work" programs.

Program Failure TechnoTalk

As I've said before, programs fail for a variety of reasons, but they usually boil down to just a few errors.

Stack Frame Size: The program doesn't check to see if it's on a 68030, which does things differently on an interrupt than a 68000.

Dumb RTE Tricks: Programs that use RTE, without checking for a non-68000 processor, as a way to make 2 lines of code into 1.

24-bit/32-bit clean: The 68000 is a 24-bit (16 megabyte) processor; the 68030 is a 32-bit (4-odd gigabyte, or billion bytes) processor. Many ST programs used the upper 8 bits of the program counter as a dumb way to store flags (personal opinion), so they break on a 68030. This is fixable with the MMU inside the 68030. Atari's fix is called 24BIT.PRG; it locks the program into a fixed 16 megabyte address space. 24BIT seems to confuse the SST; it's TT specific, so don't use it (which is unfortunate, since it meant we had to write our own). Our SST program to force the SST into 24-bit mode is called MMU24BIT, and is on the SST Release Disk.

Caution: Running MMU24BIT means that SST RAM ceases to exist; it becomes "outside the realm" of the ST's knowledge! This will continue until you restart the machine. However, it's the only way some programs will be able to run on the SST.

The classic program that requires 24-bit mode is anything written in GFA Basic™. There are doubtlessly others.

Timing: Programs that depend on timing loops, as already mentioned. Well... the whole idea of the SST is to run faster. So timing loops run faster, too. If they control something critical, like a disk access, you'll get a crash or a lockup. There's not much I can do here, except some magic I'm planning...

Glop & Other Miscellaneous Stuff: Programs that do something stupid... Malloc(0) (in short, doing a memory request, and asking for zero bytes)... programs that already work only by accident and have a bug waiting to bite them, and so forth. You'd be amazed at how much software works more or less by accident!

Editor: This from the husband who is constantly telling me that half the Mac software is riddled with zerostores! I wonder...

Heck, I taught the ST to be a Mac; that involved far, far worse than a few 68030 opcode violations! (How hard can it be?) So anyway, check out the READ.ME file on the SST Release Disk for the latest information about what works and what doesn't.

So in summary, some programs will only work in slow speed mode; some will work in medium speed mode; some will work in SST RAM high speed mode. It is possible to "tune" a program, via its PRGFLAGS, so that when you double-click on it (or otherwise run it), the program will load and run the way it needs to work. The PRGFLAGS will be saved with the program, so you'll only have to do this once. The CACHE you'll have to set by hand, and save into the desktop information file.

Other Day to Day Things

When you boot up your SST, you should always have the MAKEDSK floppy in Drive A:. Then you won't need to press any keys to abort memory tests and such, and it takes less time to bootup.

When To Initialize SST RAM

In order to "turn on" SST RAM, and do the "MaddAlt" procedure that tells TOS there's 4 or 8 megabytes extra RAM out there, you have to run the RAM initialization program. (RAMnbmm.PRG) Just *when* you do this can have *interesting* results.

First off, you don't *have* to do this every time you turn on your ST. It's not required, and I won't hate you if you don't! If you're not doing something that needs SST RAM, the heck with it - leave it off. You can turn it on anytime.

If you do decide to turn on SST RAM, you do it by double-clicking it or otherwise directly running the RAMnbmm.PRG (from command line, for instance). This is how we first initialized and tested out SST RAM.

Now here's something interesting. The GEM Desktop is just a program (admittedly, a program stored inside the ST, but still, a regular ol' GEM program) that's run after all your AUTO folder programs are run. If you initialize SST RAM *before* the Desktop starts up, many pieces of the desktop, icons, and desk accessories will load up in SST RAM! I have no idea why this is, but it works. You may get odd effects using the BLITTER if the disk icons load into SST RAM though, so it's best to leave the BLITTER turned off.

More interestingly, Q-Index, a program to measure machine speed, shows a 150% increase in speed from doing this, up to about 971% on the CPU Memory figure, if I recall correctly. What's happening is the system stack(s) are now being kept in SST RAM instead of ST RAM, and since the stacks get hammered on so much, putting them in high speed memory really helps. You'll want to experiment with this; I had mixed results with it.

Also, anytime SST RAM is initialized, the Desktop will use it as a temporary holding ground for a variety of things. This can be good or bad. Try it!

If you have the RAMnbmm program in your AUTO folder, and you do not want to initialize SST RAM, just hold down either "SHIFT" key while booting. It will disable the RAM Initializer.

MEMSTAT

One of the programs on the SST release disk will help you keep track of memory. It's called MEMSTATx.PRG, with "x" being some revision number.

MEMSTAT shows you how much memory is used, free to be used, and how much memory there is total for ST, SST, and ST with SST fastRAM. It also plots a small bar graph showing memory in use.

MEMSTAT will show you how much ST memory the ST uses "idling"; the same with SST memory. If you use programs that permanently lock up a section of memory, MEMSTAT will show them, too (say, a disk cache, RAMDisk, spell checker...). In fact, it will also tell you if you haven't switched on the SST RAM (via RAMnbmm.PRG).

Finally, MEMSTAT provides a simple check for how much SST RAM is being recognized and working. If you use Atari's CPX, clicking "General", then "Status" will show you the SST RAM size, but will call it TT RAM. (The Control Panel is also on the SST Release Disk.)

Go Have Fun!

At this point, you're pretty much on your own. You'll just need to test your programs for compatibility with plain-old-68030, SST RAM, cache, and so forth. The programs that you can run in SST RAM, set the PRGFLAGS bits for, and go for it.

Please don't be afraid to try things out on the SST. A lot of work (and I mean a *lot*) has gone into making a "safety net" under you, to prevent problems, and you cannot physically harm the ST or SST! The worst you can do is crash - so you turn the power off, then back on. Big deal! (Believe me... as possibly the person who holds the record for crashing the Mac OS, it's no biggie.) It's much better to try out a program *now*, and discover where or if it will crash in SST RAM, then when you are about to complete the final editing pass of the "Great ST Novel"!

But it's the undiscovered country; it's pushing the envelope, and that is where you find new and neat things - true Hacker Treasure.

No one knows how fast the SR-71 can go; it's "Mach 3+". No one knows how high it can go; it's "80,000 feet+". Similarly, no one knows yet just how far and fast the SST can go; only *you* can determine that!

Editor: When I would ask my Dad about the speed of the SR-71, he would always reply with the same words. The "conversation" would go something like this:

*"How fast does it go, Dad? How fast does it **really** go?"*

Dad would pause, and then simply reply, "Fast enough."

Remember to use afterburners on takeoff!

Editor: This is one of my favorite poems, because it has lots of good memories associated with it! You'll probably recognize it: years ago, they used to have it recited right before the TV station would shut down for the night.

Anyway, I thought it was appropriate for the manual.

It was written by John Gillespie Magee, Jr., a 19 year old American volunteer with the Royal Canadian Air Force, who was killed in action December 11, 1941.

High Flight

Oh, I have slipped the surly bonds of earth,
And danced the skies on laughter-silvered wings;
Sunward I've climbed and joined the tumbling mirth
Of sun-split clouds - and done a hundred things
You have not dreamed of - wheeled and soared and swung
High in the sunlit silence. Hov'ring there,
I've chased the shouting wind along and flung
My eager craft through footless halls of air.
Up, up the delirious, burning blue
I've topped the wind-swept heights with easy grace,
Where never lark, or even eagle, flew;
And, while with silent, lifting mind I've trod
The high untrespassed sanctity of space,
Put out my hand, and touched the face of God.

John Gillespie Magee, Jr.

Using the Spectre with the SST

Which Version of Spectre Do You Have?

Spectre 128™ and Spectre GCR™ share common software. Since their initial release, they have had many updates, some to fix bugs, but mostly to expand their capabilities.

Spectre 128 Software for the ST:

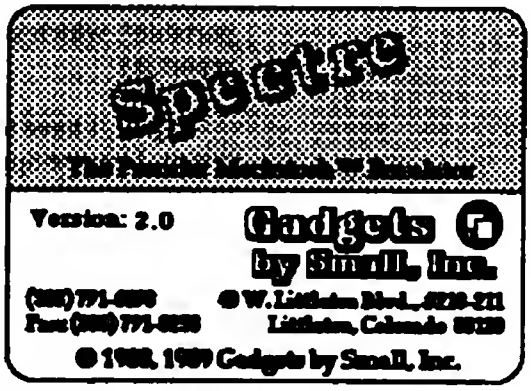
1.51: The original Spectre 128 release (before GCR existed)
1.75: Update to 1.51
1.9F: Update; gave Spectre sound for the first time!

Spectre 128 and Spectre GCR Software for the ST:

2.0: Same thing as 2.3K (some 2.3K disks were labelled "2.0" accidentally.)
2.3K: The original Spectre GCR release, with all the added software to run Mac™ GCR (Group-Coded-Recording) disk drives.
2.65: An update to 2.3K.
2.65C: A minor bug-fix to 2.65 for color monitors.

Spectre 128 or Spectre GCR Software for ST or TT:

3.0: The first Spectre 128 & GCR release that could run on the TT, plus *many* other additions. "The Best Spectre Ever".



One reason it is *very important* for you to take the two minutes to fill out and send in your Warranty/Registration card to us, is that you will receive, *free*, a good newsletter with news about Spectre and newer Spectre releases, and news about your SST. We *definitely plan* (and are writing, right now) more software for the SST to make it even more compatible and faster.

Editor: Remember, we specialize in "compatibility and speed".

Also *make sure* to read (by double-clicking it) the READ.ME file on each Spectre and SST Release Disk! They contain news that is current *as of* that release of software, which probably is more up to date than the manual! (Manual revisions take a re-layout and re-printing; READ.ME revisions take some typing and a new disk duplication run, which can happen in two days.)

With the SST, there is ***absolutely no chance*** that Spectre versions below 3.0 will operate. For example, the 68030 processor in the SST requires an "MMU" to operate in the ST hardware environment, and an MMU table was only added in Spectre 3.0, for the 68030 TT. In addition, the 68030 runs much more quickly than the 68000, and many internal "Timing Loops" fail to work correctly (slowly enough) for the GCR hardware.

Hence, if you have Spectre 2.65C or below, the first thing you need to do is to get a newer release, like Spectre 3.0

Now, Spectre 3.0 *is not* 100% "68030 perfect". Murphy's Law happened to us while developing Spectre 3.0 for the ST. In case you're not familiar with Murphy's Law (the name may be an "American-ism"):

"Anything that *can* go wrong *will* go wrong."

It happened this way.

Beta Testing

We released Newsletter #5 announcing Spectre 3.0 while Spectre was in the final Beta Test, getting last minute minor bugs reported and fixed. Orders for the 3.0 *poured* in; in fact, our mailman knows us very well, because he's carried in hundreds of letters addressed to Gadgets!

In the meantime, the final tuning of Spectre 3.0 was taking place. This is a *carefully done* process, a bit like surgery. First, I take the absolute *best* version, and mark it as "Done". I call it "casting it in concrete". Then I send it to the Beta Testers for testing. While they are busy testing, *no changes* are made; even the most trivial change to the Spectre code can have considerable (sometimes disastrous!) results. The Beta Testers come back to me with a list of bugs and suggestions if there's anything wrong; I then fix the code, once again "cast it in concrete", and send it out to the Beta Testers for testing.

We all did this 18 times with Spectre 3.0; it literally went to Beta-18 version. But don't feel bad; I have seen some well-known Mac products go to Beta-45 versions! (We are Beta Testers for quite a few well-known Macintosh software companies, because if their software runs on the Atari with no "zerostores" or other errors, it is "clean" and "machine portable", which is what they want. If there is an error, we can *easily* find it for them and report it.)

Again, I have to emphasize that I *do not* change a software version while it is being tested. Imagine Spectre as a spiderweb of code, everything touching everything else; tug on one place, make one change, and ***the whole thing vibrates.***

For example, there was Beta-18. We received a frantic call late at night from one of our most devoted Beta Testers who had re-checked literally hundreds of Mac programs with Beta-17. He found that one program, recently updated, which is considered absolutely vital to have with the Mac in Desktop Publishing (I cannot name it here) *failed*. I looked into the problem. The program was generating a Nil Pointer (a programming problem - now you know why I can't name the product!), which equals zero (0). Then, the program was *reading* from the "pointer", which is the sort of thing most programs do; they build up "data structures" composed of many different data items, point to the beginning of the structure, then offset down a little bit to the particular item they want. This program then took what it read and fed it to the Font Manager™ of the Mac as data. The problem was that zero'ed pointer - instead of looking at a data structure inside the program somewhere, the program was looking at location $70 in memory!

What they were *trying* to do was set up a data structure, read one item out of it relating to fonts, and send it to the Font Manager. What they *actually did* was read location $70 and

send it to the Font Manager. Location $70 is *always* the "Exception Vector" that handles "Vertical Blank", something that happens 70 times a second in monochrome, 60 times a second in color. Location 70 contained an address in the Spectre code that did our VBL work.

In Version 2.65 of Spectre, and in early Beta versions of Spectre 3.0, it just so happened, by *total accident*, that the location ended up being a number that was harmless to the Font Manager. (It turns out back then the Font Manager had already been initialized once, and was rejecting the number given it because it did not make sense). In ordinary words, the program was working on Spectre by *accident*.

In Spectre 3.0, however, the Spectre program had gotten much bigger (because of all the 68030 support we had just added to it!) and various routines, like the VBL routine, had moved around. Location $70 was thus different. When the newly released version of the Page Layout program got the value accidentally read from $70, it went crazy and the system crashed.

Finding out what's going on is the hardest part of debugging (re: Heisenberg's Uncertainty Principle). The solution? I just moved the VBL routine around inside the Spectre code (it's a separate module, anyway) so that its address, the number that would be fed to the Font Manager, was happy. This cured the problem.

That's what Beta Test is like - a *carefully controlled test of software*. You must change only one thing at a time, since changing two things can cause an interaction you don't want, and can't foresee, and can break just one of thousands of Mac programs!

Anyway, back to 68030 support for the TT and SST.

68030 Support in Spectre 3.0

We were at the end of the Beta Test for the ST version of 3.0 when suddenly, unexpectedly, an Atari TT arrived.

Believe it or not, this was not good news for us. Why? ***Because 3.0 was almost ready to go***, and adding the stuff to support the 68030 in the TT would substantially change the program. In fact, we'd practically have to go back to Beta Version 1.

Because of the time it takes for Beta Testing and finding and fixing some particularly subtle bugs, we were already receiving phone calls and FAXes and letters saying, "Hey! Where is my copy of Spectre 3.0? I ordered it some time ago." So we were under pressure to ship the ST-only version of Spectre 3.0.

There were several alternatives.

Alternative #1: We could release Spectre 3.0 when it finished Beta as an ST-only product. This would irritate TT users who were already asking for a TT compatible version of Spectre.

Alternative #2: We could add *limited* 68030 support for the TT; stuff that would not take too much time to add (since we didn't have time), and return to Beta Testing. Then we could ship with maybe two to four weeks added delay.

Limited 68030 TT support meant we made Spectre 3.0 *just able to work* on the TT. It *does not* mean we support the SCSI internal hard disk, TT Fast RAM (another totally new thing), or the TT Localtalk port.

Alternative #3: As the last option, we could take several months, add full 68030 SST and TT support, irritate our ST customers *a lot* while they waited on something they had no use for (68030 based stuff - you have to remember that in January 1991 there were *very*

few TT's out there, and no SST's). Remember, we were *already* getting a lot of calls about "Where is 3.0? "

So, we decided on Alternative #2 - limited 68030 (TT) support. We worked *very* fast, in near 24-hour shifts, getting enough code into Spectre to support the 68030 (for instance, an MMU table), making *sure* it worked on the TT with Spectre and Mac disks, and making *sure* it worked with both a TT "regular" Multisync color monitor and with the 2-page monochrome monitor, called TTM-194 in the USA. Really, there was not much choice, so we made the best of it.

It Is NOT a Bug!

I mention this *because* many people seem to think that Spectre 3.0 not working with TT SCSI or the TT LocalTalk port is a program bug; well, ***it isn't***. There is no programming in Spectre 3.0 to support those things, because 3.0 was not meant to be full TT or SST 68030 support. We are going to put as much TT and SST support as we can into the next release of Spectre, then reserve anything that takes a *lot* of time for later versions.

Please realize that the *primary* function of Spectre 3.1 is to support Apple's System 7™, which *does not work* with Spectre 3.0 (or any other version); many of our customers want to run System 7.

Spectre 3.0 and the SST

Spectre 3.0 runs fine with the SST, ***if you keep a few things in mind.***

Maximum Oscillator Speed

First, ***DO NOT run above 33.3333 MHz!*** We cannot guarantee the reliability of Spectre 3.0 if you run at more that 33.3333 MHz.

When Spectre 3.0 was released, it was "tuned" so it could run at the maximum TT speed, 32 MHz with the 68030's caches ON. At the time, we had no idea that the SST could run faster, like say, 40 MHz, so Spectre 3.0 was *not set up* for the SST's potential of "warp drive". There are definitely sections of the Spectre program that ***will fail*** if you go faster than 33.3333 MHz; so that is the fastest oscillator speed that *we* recommend, at least right now.

Anyway, in Beta Testing at 40 MHz, Spectre 3.0 would get random crashes. Sometimes Spectre would run for some time (and darn fast); sometimes it only ran for a few seconds. In other words, ***Spectre 3.0 is not reliable at speeds over 33.3333 MHz.***

Caution: YOU MUST NOT try to run either GCRTEST or SPECTRE in SST RAM, by setting their "program flags". That will absolutely, positively prevent those programs from working.

This also applies to TRANSVERTER or LAUNCH or ANYTHING on the Spectre 3.0 program disk.

Also, because the floppy disk timing loops in particular had to be ultra-fine-tuned; floppy disks do not work *at all* at 40 MHz; caches on or off doesn't even matter. It is extremely difficult to write code that works ***both*** at 33 MHz and on a stock 8 MHz ST, believe me, I know; especially when there is not even CPU room and time for a tunable timing loop at 8 MHz!

Worse, the Spectre hard disk code was also tuned to the TT speed (basically 33 MHz), and is unreliable at 40 MHz. Two of our Beta Testers ***DAMAGED DATA ON THEIR HARD DISK*** (and had to reformat) by running Spectre at 40 MHz.

I expect to fix these timing bugs when I have time to look over the Spectre code in not such a terrible hurry as with Spectre 3.0. For example, the routine that shuts off keyboard interrupts is known to be speed-sensitive, as are the floppy disk, hard disk, and other routines. We did our best to fix these things in the SST software, but we couldn't fix everything. The future release of Spectre will be much more 68030 compatible, and much less "tuned" to a maximum MHz.

Floppy Disks

Sometimes the floppy disks do not work when Spectre is run at it's top speed, 33.3333 MHz with caches ON (and SST RAM OFF). This is one of those timing-loop problems where the hardware is being run too quickly and has a problem. You can *very often* solve this simply by *turning OFF* the "Caches" using the right-most Spectre Menu. Just switch Off "Instruction Cache", "Data Cache", and "Burst Mode".

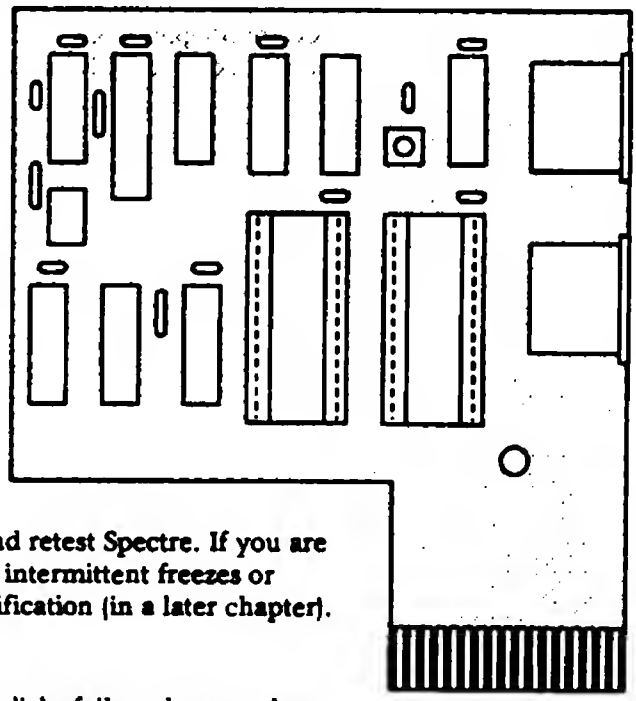
This *will* result in your Spectre speed falling off, of course. You *only* need to do this, according to our Beta Testers, if you plan on using floppy disks; if you run on the hard disk, you can leave the Caches ON to considerably increase your speed. (Note: Spectre 3.0 only uses ST RAM, not SST/TT Fast RAM. Hence Burst Mode is impossible, since ST RAM does not use it.)

If you still have floppy problems with Caches OFF, you can try using SPECTRE format floppy disks, which are written and read using normal MFM (e.g., the ST's regular disk style). You may find that they are stable, whereas the ultra-speed-sensitive Mac GCR disks may be unstable.

GCR Cartridge Timing and the SST

Sometimes the GCR hardware itself just cannot deal with the SST speed (especially if you are running at 33.3333 MHz). The GCR was designed for the ST cartridge port long before the 68030 TT or SST were more than a gleam in their designer's eye.

Sometimes the SST's timing (your oscillator MHz value) is sufficiently different from the ST's timing that the GCR cartridge does not work all the time. This can vary from a failure once a day to almost continual errors. It can show up as ROMS ARE BAD messages that prevent Spectre startup, or show up as floppy disks not working *in Mac Mode*, or floppy disks not working *at all*.



There is a hardware modification that can be made to the GCR cartridge which will help this condition considerably! (It adjusts the GCR hardware to handle 68030 SST and TT timing, but also keeps it timed okay for the unaccelerated ST.)

But there are *other things* that can cause identical symptoms that you should check for *first*, before you "cut and paste" your GCR. (And besides, they are easy to check for.)

If your GCR suddenly quits working and it was working fine before you installed the SST, do a new "SAVE SETTINGS" to make sure you are not confusing the GCR. Then run Spectre to see if it's fixed.

If not, try turning OFF the caches in the Spectre MENU, and retest Spectre. If you are still getting non-floppy related errors, like ROMS ARE BAD, or intermittent freezes or crashes, you will probably need to do the GCR Hardware Modification (in a later chapter).

Floppy Disk Problems

I have already mentioned that the Caches can make floppy disks fail; make sure they are OFF before you test your GCR cartridge, preferably first with GCRTEST (which tests the ability of your cartridge to format, write, and read GCR/Mac format disks).

Note: This means you *must* turn off the "CACHE" entry from the ST desktop before running GCRTEST, because you are running in ST Mode! Otherwise your GCR may have no problems, but you will still be getting errors in Mac Mode because the Cache entries in the Spectre right-most menu are still "ON".

Also, make sure that *floppies in general are working!*

One problem that stands out "In Particular" is if you have your video monitor right on top of the ST. The monitors tend to emit large magnetic fields and electro-magnetic interference - and floppies are electromagnetic devices! I fixed my own ST of this problem *just by moving the monitor*, where its interference to the floppy drive was not so strong. (It was randomly failing in both ST and Spectre (Mac) modes).

Incidentally, the internal and external floppy drives *use the same wiring*. This means that while your external floppy may be quite a distance from the monitor, its *wires* can pass right close to the monitor, and can pick up enough interference to stop working. (The interference "drowns out" the floppy signal just enough to thoroughly confuse GCR.)

One good way to test this is with GCRTEST. First, test Spectre mode floppies; heck, these are just ST-type floppies (10 sector, though), not much special about them, formatted by the ST. If that test fails, you have floppy problems that have nothing to do with the GCR cartridge.

Then, use GCRTEST to test Mac mode floppies, which exercises the GCR cartridge quite a bit. (In fact, formatting a Mac disk is the hardest test of all for the GCR hardware and software!)

Also, bear in mind that some GCR's in some Atari machines require "tuning" via the potentiometer to make them work. (There is a lot more about this problem in the Spectre GCR manual.) You could simply have the resonant-spike problem that the tuning overcomes.

If you find out that floppies in general work in ST mode ***AND*** GCRTEST says that Spectre floppies work, and Mac/GCR floppies do not work, then it is time to look into modifying your GCR hardware.

The Future of the SST

There's an old saying in the computer industry that goes like this:

"At some point in the life of a project, it's necessary to shoot the engineers and put the thing into production!"

Editor: Shoot the engineers and the software team!

Other Editor: And the Editor!

The SST, as-is, is very fast, very compatible, and very inexpensive for what it does. In terms of Atari's machines, it's pretty comparable to the TT in many ways; the memory structure is similar (Atari calls SST RAM "TT RAM"); the TOS is similar (your TOS 2.06 is a derivative of TT TOS 3.0x; Atari sorta removed the TT-only stuff from TOS 3.0x to make 2.06), and certainly, we've got the speed, too. Depending on how fast you run your SST, it will either be slower or faster than the TT.

However, we've also got some things that the TT hasn't. For instance, even if you forget to run MMU24BIT, you'll likely find that 24-bit programs work fine! This is no accident, this is *our* hardware and software design.

As our Beta Testers keep telling us, "It's good enough as is! Ship it, will you?!? It's fast and compatible. Quit polishing the polish!"

So we're (ahem) shipping.

What we're working on now is software to make the SST more "seamless", to make working with it easier, more intuitive, and less likely to crash. As we did with Spectre 128 and Spectre GCR, we're looking at why programs fail and how to fix them, and with each SST software version, more and more stuff will work, and more features will be added.

For instance, wouldn't it be great if the SST knew to switch to 24-bit mode automatically for a GFA-Basic program, and to switch back into 32-bit mode when the program finished?

Wouldn't it be great if programs "saw" 12 megabytes of ST RAM, without knowing that 8 megabytes of it is SST RAM?

It sure would be helpful if programs that used delay loops had some extra help slowing down when they hit hardware, to prevent crashes.

And so on.

That's the sort of thing we're developing now. For instance, part of the 12 meg software is already completed. That was among the first things I wrote... and if some aftermarket hard disk software "obeyed the rules" better, it would be on your SST disk right now! But, I'm long resigned to fixing other people's software.

Editor: Not to mention having loads of experience at it!

However, if we wait for all these neat software things to be finished and perfect, it'll be 1999 before we ship... and here we've got hundreds of tested SST's decorating our office (and living room, and dining room, and family room, and...)

I'm afraid it's inevitable. We have to ship them before any more cat hair gets into the boxes! You see, we have a cat that positively dotes on boxes, and sleeps in them; this is wintertime in Denver, a very dry climate, and the static electricity in our cat's fur sparkles and snaps...

So, out they go, kicking and screaming into the cold, cruel world.

Be sure to send us your Registration Card, so you can find out all about all those future SST Updates!

Gadgets SST Support

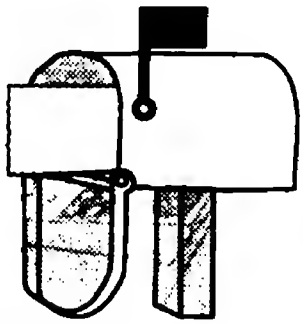
We try to support our users to the best of our ability, but in order to do so, we need *your* help! The best way is to send your SST registration card directly to *us* (you can send it by FAX or mail). This will *ensure* that you get on our SST Customer mailing list. We'll be letting you know about software upgrades we're working on and we'll also send you the latest copy of our newsletter, the "Gadgets News-Herald-Update-Enquirer"!

And, if you want to, you can be part of the SST evolutionary process. When you find a bug, let us know!! We can't fix bugs we don't know about, and by necessity, I don't have the time or money to look over and test *all* the ST software out there; I'm usually off writing SST or Spectre code, or a Current Notes article, or telling our kids a bedtime story.

Editor: With Whitney and Christopher and a ghost named "Samuel Clemens".

Very often, a bug will show us a *class* of problem (like MALLOC(0)), and we can fix it. Then, not only your program, but also many others that do the same thing, will start working.

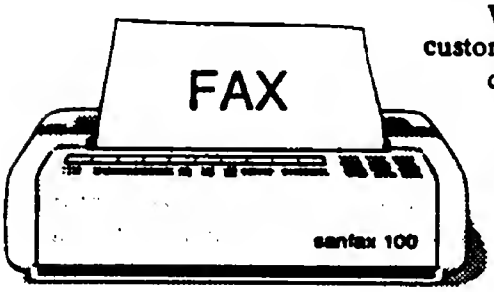
We've seen this many, many times with Macintosh™ software and Spectre. For example, the zero read bug which was killing Microsoft Word 3.02™ was also what made MultiFinder™ so flaky - we fixed Word, and MultiFinder also started working right.



So, when it comes time to get questions answered, report a problem you are having, or a bug, or whatever, you've got several choices:

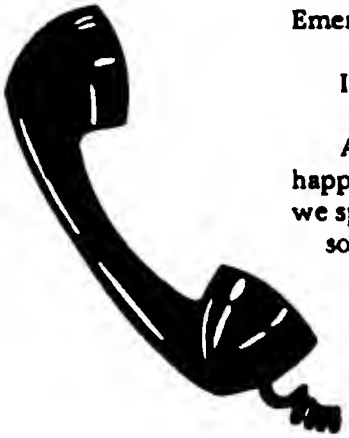
- You can drop us a letter. This takes a while to turn around, but is mighty inexpensive. It also allows you to enclose a disk, showing us the problem, so we can see it, and possibly fix it for the next software release.

- You can send us a FAX at (303) 791-0253. We have a handy dandy little device that just loves to spit out paper, and it's available 24 hours a day. (Unless someone else "beat you to it" and the line is busy, or unless Jamie has decided to rewire the phone lines.)



We find FAX is the best way for us to support our customers, as the questions and problems are written down. Generally, we just FAX back the original, with the questions answered, so please try to leave space for writing answers.

Please be patient concerning your reply; we're quite busy. Also, make sure there is a *legible* FAX number or mailing address, so we



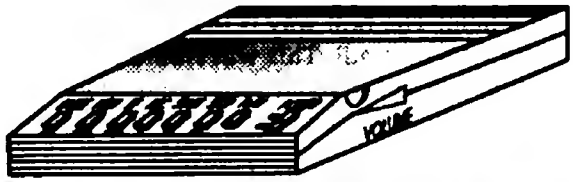
can send an answer *back* to you.

- You can call us on the phone, at (303) 791-6098. We answer the phone Monday, Wednesday, and Friday from 8:30 AM to 2:30 PM, Mountain Time, unless it's a Holiday or Emergency.

It's a long distance call for you, almost certainly. Denver is *not* an oasis of Atari users.

Also, the line is probably busy because we're dealing with someone else. This happens. There are *a lot* of someones. You have to understand that when we take one call, we spend X minutes satisfying one person (maybe); when we put a fix or upgrade into the software, we make a few thousand users happy. Where do you think we should draw the line? We enjoy talking with people, but there's only a certain number of five-minute periods each day.

The most depressing fact is that a lot of tech calls are questions that are already covered in the manual.



- You can talk to our answering machine. Because there are just the two of us here, we have to concentrate ourselves; our phone availability is *limited*. Our answering machine is available for messages after hours, and is pretty reliable (although I *do* discover "mystery messages" occasionally, due to "help" from Jamie, the intrepid almost 4 year old!).

- You can contact us online, on CompuServe, GEnie, or Usenet.

The GEnie Gadgets Round Table is the "home base" for Gadgets support; all of the latest software and relevant discussions (well, mostly relevant) are there. We recommend that you check it out. Type "gadgets" to get to our area, or send a note to "davesmall".

On CompuServe, you can find discussions about Gadgets products if you type "go atariven" to get to the Atari Vender Support areas. Or you can send email to 76004,2136.

To get support over Internet or Usenet, try sending a message to dsmall@well.sf.ca.us, and cross your fingers. (I know, it makes it hard to type!) For some obscure reason, I seem to only get about half the messages people send to me, so be patient.

If there's a Gadgets area on the system you dial into, you probably will be able to find a topic that answers your questions, right there and ready to look at, from some previous user who asked the same question. So take a moment and browse around.

SST Hardware Options

68030 Microprocessor

The SST requires a 68030 microprocessor (and also an oscillator; see below) to operate at all. These required microprocessors are only manufactured by Motorola. The SST can use either the 16 MHz version, the 25 MHz version, the 33 MHz version, or the 50 MHz version. (Naturally the price goes up almost as fast as the speed!) Below is a chart with the available MHz, corresponding part number, and oscillator speeds the 68030 can safely operate at.

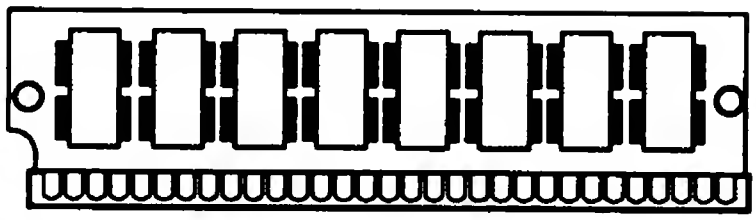
µp	Part Number	Oscillator Range
16 MHz	MC68030RC-16	18 to 20 MHz
25 MHz	MC68030RC-25	20 to 28 MHz
33 MHz	MC68030RC-33	26 to 38 MHz
50 MHz	MC68030RC-50	33 to 40 MHz (Note: SST design limit is 40 MHz)

Coprocessor (Floating Point Unit)

The SST has a socket for an FPU or coprocessor, but an FPU is not required for operation. There are 2 models which work with the SST; the 68881 or 68882, which are also only manufactured by Motorola. (The 68882 is faster then the 68881 at the same speed, but either will work fine.) Please note that for optimal SST operation it is best if the speed of the FPU and 68030 are the same; also, the FPU's do not "overclock" as well as the CPU's. Below is a chart listing part numbers.

FPU	Part Number
33 MHz 68881	MC68881RC-33
16 MHz 68882	MC68882RC-16
33 MHz 68882	MC68882RC-33

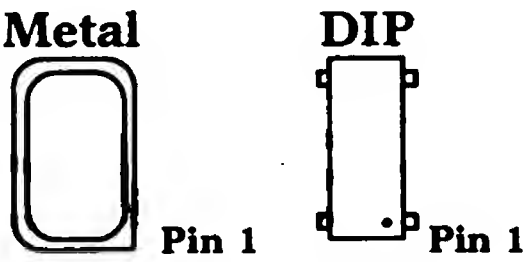
SIMMs



The SST has 8 low profile SIMM sockets. The SST can *operate* without any SIMMs (SST fastRAM), but you won't get ear-bleeding speed without SIMMs installed. You must install either 4 or 8 SIMMs, and the specs are basically 1 megabyte, page mode. Other than that you have a choice of 1x8 (Mac style) or 1x9 (IBM style), and the speed (we use either 80 or 70 nanosecond). We highly recommend that you use 1x8 SIMMs at 70 nanoseconds. MacWeek magazine has numerous sources for SIMMs listed each week. Note that the last digit before any non-numeric must be a 0 (which is what specifies page mode). Here is a representative part

number for a 1 megabyte, 1x8, 70 nanosecond, page mode SIMM: KM41C1000BJ-7. The "1000" means it is one megabyte "page mode".

Oscillators



The SST comes with two oscillators: a 20 MHz and a 33.3333 MHz. As you can see from the 68030 chart above, you don't have to match the speed of the oscillator with the speed of the microprocessor. The SST was designed asynchronously, which basically means you can plug in any value oscillator from 18 to 40 mhz. The upper limit is restricted to 40 mhz because of the DRAM controller (there isn't a 50 mhz version available). The package type is the standard 4 pins, which fits a 14 pin socket. The representative part number for a 33.3333 mhz CMOS oscillator is: SG51KT33.3333MC.

Where to Order the Parts

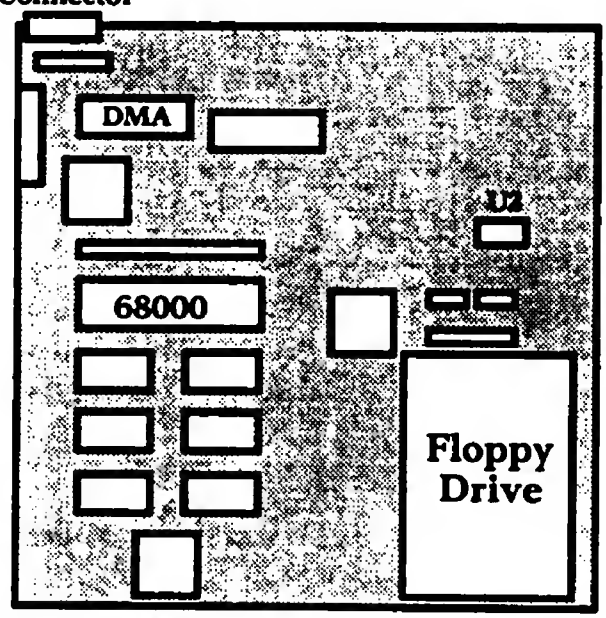
Digi-Key Corporation
701 Brooks Ave. South
PO Box 677
Thief River Falls, MN 56701-0677
Phone: (800) 344-4539
Fax: (218) 681-3380

Mouser Electronics
2401 Highway 287 North
Mansfield, TX 76063
Phone: (800) 346-6873, (817) 483-4422
Fax: (817) 483-0931

George's Hard Disk Fix

Sometimes an SST won't boot with a hard disk attached, won't boot with a particular hard disk attached, or won't boot with more than one hard disk attached. This is because when running at higher speeds, the SST is particularly noise sensitive. Most of the lines coming into the SST are buffered and cause no problem, but the RESET line is bidirectional, so it can't be buffered on the SST.

This problem, if it happens, only shows up with hard disks attached because the RESET line runs out to the DMA port. The hard disks you have may not be buffered; if they aren't, the RESET line can pick up a lot of noise, which will confuse the 68030. This problem can also show up as DMA failures on a regular unaccelerated ST.



Illumination 28 - Location of U2 and DMA Connector

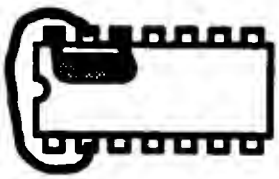
IMPORTANT: Read the entire procedure before beginning. If any part of it doesn't make sense, don't perform the modification! Find someone who is experienced with electronics to do it for you. Use this information at your own risk! Gadgets by Small, Inc. assumes NO responsibility for any damage or expenses arising from the use of this information.

We highly recommend that you have your Atari Dealer do this modification; if you don't know what you are doing, you could ruin your ST and end up with a major repair bill!

If you are determined to fix this ST problem yourself, you'll need some "wire wrap" wire, wire cutters, solder, a drill with a tiny drill bit installed or an X-acto knife, and a 1K ohm 1/4 watt resistor.

1. Take apart your Mega, and remove the floppy drive, power supply, and SST if it is installed.

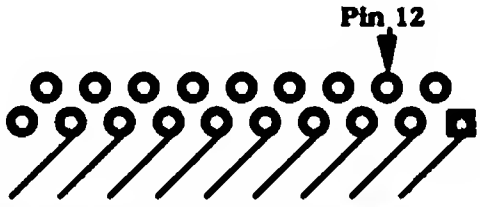
2. Find the 7407 chip labeled "U2" on the Atari ST motherboard. It is located under where the power supply was. (Illumination 28.)


Illumination 29 - U2 Changes

3. Solder a short piece of wire from pin 2 of U2 to pin 13 of U2. (Illumination 29.)

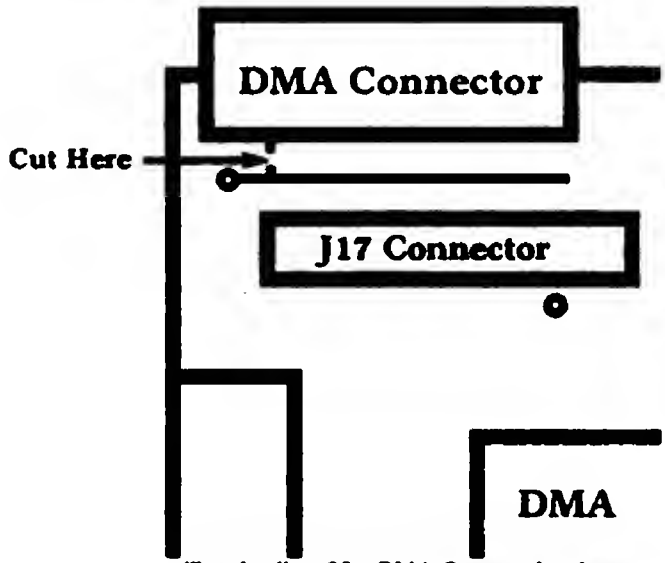
4. Then solder a 1K resistor to U2 between pin 14 and pin 12. (Illumination 29.)

5. Now you need to solder a wire from pin 12 of U2 (there is already a resistor soldered to that pin) to pin 12 of the DMA Connector on the back left corner of the ST motherboard. The easiest way to get the wire to pin 12 of the DMA connector is to solder it to the underside of the ST motherboard. (Illumination 28 and 30.)



Illumination 30 - Pin 12 Under the DMA Connector

6. On the top of the ST motherboard next to the DMA Connector is a trace which looks kind of like an upside down "T". You need to cut the connection between the DMA Connector and the top part of the "T". (Illumination 31.) The easiest way is to is a tiny drill; but be very careful, and do not drill very deep! You can also use an X-acto knife to cut the trace. Make sure that you cut the trace all the way through, and don't accidently cut any other traces.

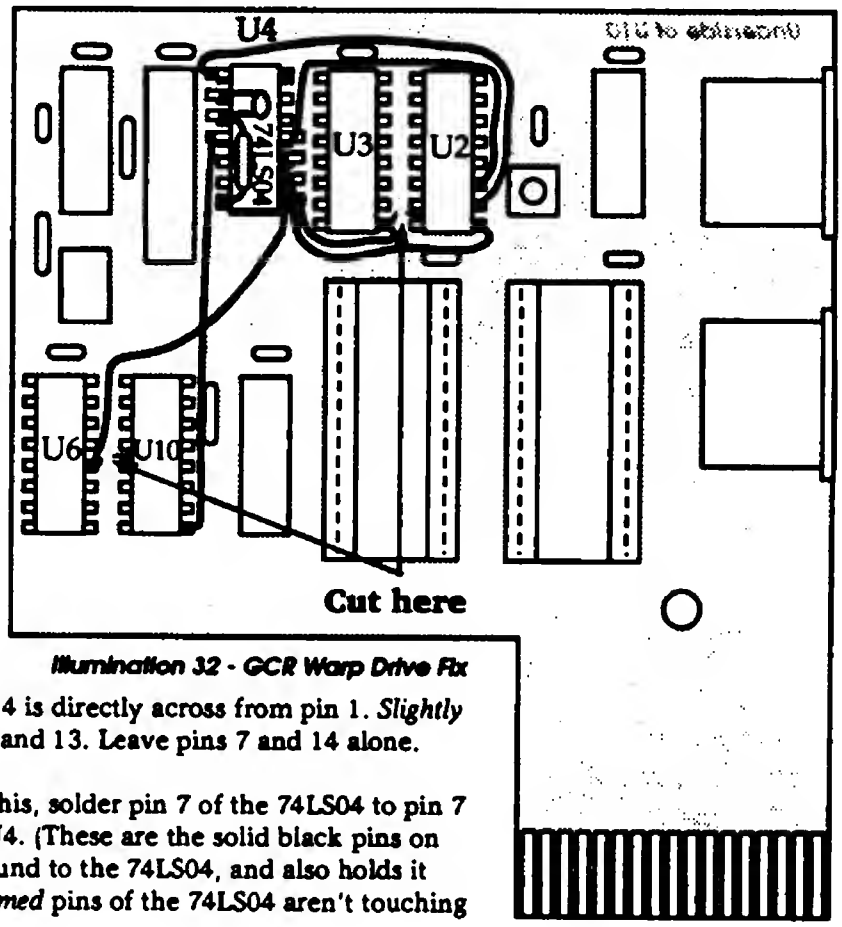


Illumination 31 - DMA Connector Area

Modifying the GCR for Warp Drive

Doing this yourself is not something to attempt if you are a novice with soldering! Parts can be damaged by static, excessive heat, or in truth, for no darn reason at all sometimes (it seems like it, anyway). This modification *will not* affect GCR operation on the ST; in fact, it seems to make the GCR run slightly better, because it cuts down noise.

IMPORTANT: Read the entire procedure before beginning. If any part of it doesn't make sense, don't perform the modification! Find someone who is experienced with electronics to do it for you, like an Atari Dealer. Use this information at your own risk! Gadgets by Small, Inc. assumes NO responsibility for any damage or expenses arising from the use of this information.



Illumination 32 - GCR Warp Drive Fix

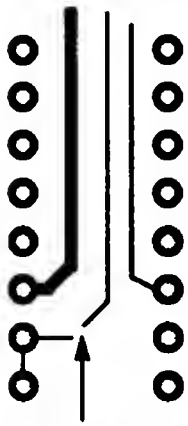
You will need some "wire wrap" wire, a 100 pf capacitor (35 volt ceramic works the best), a 1K ohm 1/4 watt resistor, and a 74LS04 chip, plus tools. You can get these parts from your local Radio Shack.

1. On the 74LS04 chip, there will be either a dimple or half-moon on one end of the chip, with a dot to designate pin 1. Start counting pin numbers from the upper left (pin 1) and down around in a "U" shape. Pin 14 is directly across from pin 1. *Slightly trim* pins 1 - 6 and 8 - 13. Cut off pins 5, 6, 12 and 13. Leave pins 7 and 14 alone.

2. Mount the 74LS04 on top of U4; to do this, solder pin 7 of the 74LS04 to pin 7 of U4, and pin 14 of the 74LS04 to pin 14 of U4. (These are the solid black pins on Illumination 32.) This supplies power and ground to the 74LS04, and also holds it securely. Also, make sure that the *slightly trimmed* pins of the 74LS04 aren't touching anything.

3. Cut the trace going from pin 9 of U3 to pin 9 of U2. Also cut the trace going to pin 12 of U6. You can check continuity with a meter to make sure the traces are cut all the way through. (Illumination 32.)

4. Solder a short piece of wire from pin 9 of the 74LS04 to pin 9 of U2. Next, solder a piece of wire from pin 8 of the 74LS04 to pin 9 of U3. Then solder a wire from pin 10 of the 74LS04 to pin 12 of U6, and another wire from pin 11 of the 74LS04 to pin 11 of U2.



Illumination 33 - Underside of U10

5. Trim the legs of the 100 pf capacitor and solder it between pins 2 and 3 of the 74LS04. One leg should be soldered to pin 2, and the other to pin 3.

6. Trim the legs of the 1K resistor and solder it between pins 3 and 7 of the 74LS04.

7. Solder a wire from pin 1 of the 74LS04 to pin 10 of U2.

8. Find U10. On the underside of the board, under U10, cut the trace going to pin 10 of U10. (Illumination 33.)

9. Then solder a wire from pin 4 of the 74LS04 to pin 10 of U10.

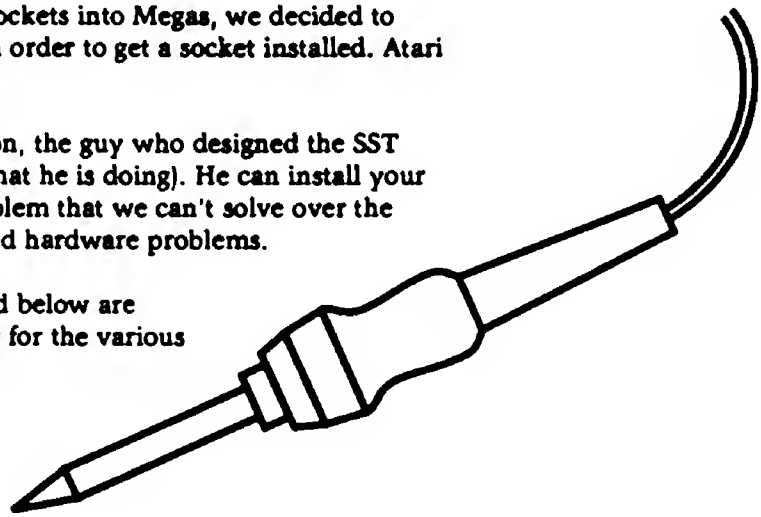
10. Check to make sure you connected and cut everything correctly, and haven't shorted anything together, and that you don't have any "cold solder joints".

11. Put your GCR back in its case, hook it up to your Atari again, and run Mac Mode.

SST Installation & Service

Since we do not have the resources to install sockets into Megas, we decided to provide a list of places you can contact yourself, in order to get a socket installed. Atari Service Centers or Dealers can also help you.

Our first recommendation is George Richardson, the guy who designed the SST hardware (so you can be assured that he knows what he is doing). He can install your socket into your Mega, or if you have a wierd problem that we can't solve over the phone, he can help diagnose and fix your ST related hardware problems.



Please be aware that *none* of these people listed below are employees of Gadgets, and they ***set their own fees*** for the various services they perform.

George Richardson
Merlin Group, Inc.
96 Hoyt Street
Kearny, NJ 07032-3311
Phone: (201) 998-4441
Fax: (201) 998-0932

Dave Troy
Toad Computers
556 Baltimore Annapolis Blvd.
Severna Park, MD 21146-3818
Phone: (301) 544-6943
Fax: (301) 544-1329

Bruce Carso
B&C Computervision
2730 Scott Blvd.
Santa Clara, CA 95050
Phone: 408-986-9960
Fax: 408-986-9968

Brad Davis
Dream Park Development
1390 So 1100 E #104
Salt Lake City, UT 84105
Phone: 801-487-8648
Fax: 801-466-2541

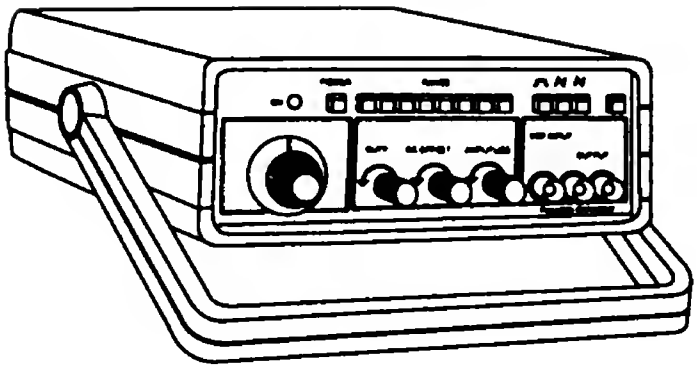
SST Installation & Service

Donald DeRosa
822 Barlow Street
Philadelphia, PA 19116
Phone: 215-969-4622
Fax: 215-673-7005

Karl Hamacher-Gatzweiler
HG Computers
Giselastraße 9
5100 Aachen Germany
Phone: 49-241-603252
Fax: 49-241-603242

Francis Fima
Clavius
19 rue Houdon
75018 Paris France
Phone: 33-1-4262-9019
Fax: 33-1-4262-9585

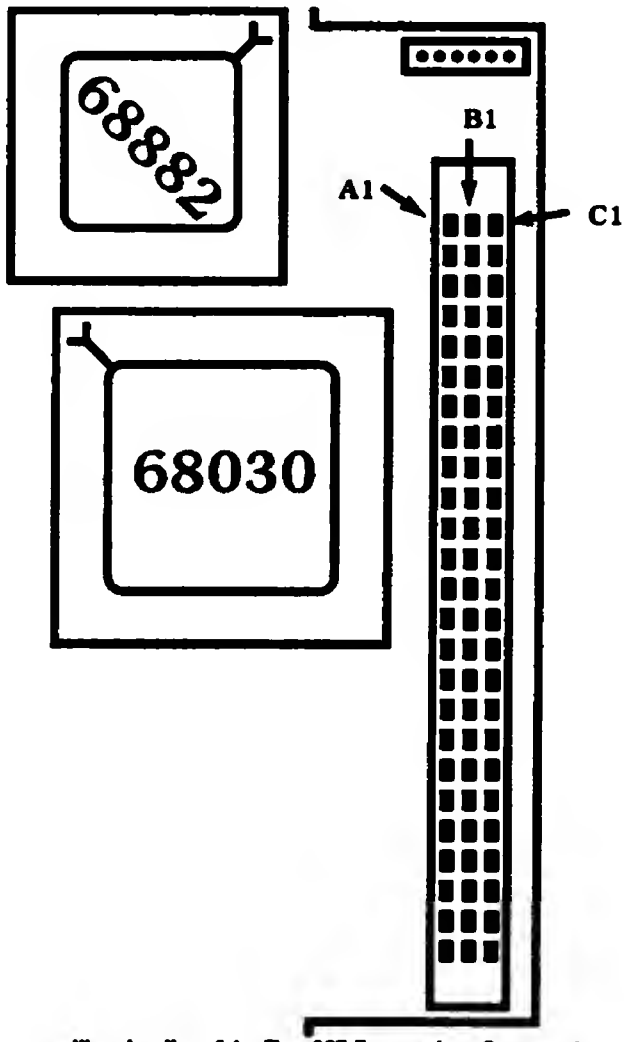
Dinologics • Stephan Muhs
Wilhelmstr. 51 • 5 Köln 60
Telefon 02 21 / 7 39 34 84
Telefax 02 21 / 7 39 01 27



SST Expansion Connector Specifications

The SST Expansion Connector is "processor direct", which means you can "talk" directly to the 68030. There are 3 rows of 32 pins on the connector, labeled A, B, and C. Illumination 34 shows the locations of the rows and pin number 1 of each row.

Pin #	Signal name
A1	- A28 •
A2	- A25 •
A3	- A22 • ••
A4	- A19 • ••
A5	- A16 • ••
A6	- A13 • ••
A7	- A10 • ••
A8	- A7 • ••
A9	- A4 • ••
A10	- A1 • ••
A11	- /DSACK1 •
A12	- /IPL1 • •• (See Note 5)
A13	- FC1 • ••
A14	- SIZ0 •
A15	- /BR •
A16	- /BG •
A17	- /DS •
A18	- /CBREQ •
A19	- /BERR • ••
A20	- +5 Volts
A21	- Ground
A22	- D1 • ••
A23	- D4 • ••
A24	- D7 • ••
A25	- D10 • ••
A26	- D13 • ••
A27	- D16 •
A28	- D19 •
A29	- D22 •
A30	- D25 •
A31	- D28 •
A32	- D31 •
B1	- A30 •
B2	- A27 •
B3	- A24 •
B4	- A21 • ••



Illumination 34 - The SST Expansion Connector

B5 - A18 * **
B6 - A15 * **
B7 - A12 * **
B8 - A9 * **
B9 - A6 * **
B10 - A3 * **
B11 - A0 * **
B12 - /IPL2 * ** (See Note 5)
B13 - FC2 * **
B14 - SIZ1 *
B15 - /BGACK *
B16 - /CIIN *
B17 - /AS *
B18 - /CBACK *
B19 - /HALT * **
B20 - +5 volts
B21 - Ground
B22 - D0 * **
B23 - D3 * **
B24 - D6 * **
B25 - D9 * **
B26 - D12 * **
B27 - D15 * **
B28 - D18 *
B29 - D21 *
B30 - D24 *
B31 - D27 *
B32 - D30 *

C1 - A31 *
C2 - A28 *
C3 - A25 *
C4 - A22 * **
C5 - A19 * **
C6 - A16 * **
C7 - A13 * **
C8 - A10 * **
C9 - A7 * **
C10 - A4 * **
C11 - A1 * **
C12 - /DSACK0 *
C13 - /IPL0 * **(See Note 5)
C14 - FC0 * **
C15 - No Connection, use it for what you want.
C16 - /EXST: This is the buffered version of the 68030 /STERM signal
C17 - R/W * **
C18 - 68030 clock, 16mhz to 40mhz * (See Note 4)
C19 - /RESET * **
C20 - +5 Volts
C21 - +5 Volts
C22 - Ground
C23 - D2 * **
C24 - D5 * **
C25 - D8 * **
C26 - D11 * **
C27 - D14 * **
C28 - D17 *
C29 - D20 *

C30 - D23 *
C31 - D26 *
C32 - D29 *

* These signals are connected directly to the 68030.
For a complete explanation, see the 68030 data book from Motorola.

** These signals connect through to the ST computer bus.

Technical Notes:
1) Addresses in the range $2000000-$2FFFFFF are free for use by expansion cards.

2) /EXST is the buffered version of the 68030 signal /STERM. It is subject to the same restraints, but imposes a 15ns propogation delay through a GAL.

3) Any card using address lines above A25 must provide some means of preventing accidental selection of addresses above $FFFFFF by the ST DMA circuitry when the 68030 has given up bus possesion. This can be done by insuring that the card does not respond unless the 68030 /AS line is asserted. The ST bus cannot assert the 68030 /AS line.

4) The expansion card designer should be aware that the 68030 clock frequency is not fixed from machine to machine. It can vary from 18 MHz to over 40 MHz. If the expansion card can't handle this range, the designer must either provide his own clock and run the card asynchronously, or enable the user to set the card up in some fashion for the correct frequency. Remember, on an SST *any* frequency between 18 MHz and 40 MHz is not only possible, but given the user ability to plug in oscillators, it's likely.

5) The interrupt lines on the Atari Mega ST are not open collector. This means that they may not be useed to generate an interrupt, only as a reference. If interrupt capability is required, a wire must be connected to the input of the priority controller on the Mega motherboard.

SST Cockpit Data Structures

Technical Note: This chapter is intended only for people with pocket-protectors, knowledge of "C" and assembly language, pony-tails tied back with a rubber band, and various other signs of hackerdom.

The ST after TOS 1.6 featured a data structure called the "Cookie Jar". In it were many pairs of two 32-bit Longs. The first was 4 ASCII bytes identifying the "cookie"; the second was the "cookie", which was either simple data, a pointer to data, or a pointer to a structure or program.

Editor: First, "Byte", then, "Nybble", and now: "Cookie"!! Sheeesh!

TOS uses the Cookie Jar to store things that programs need to know and be able to easily look up in a standard manner, such as what machines they are running on (ST/TT/whatever), which microprocessor they have, if they have an FPU, etc. However, TOS also allows us programmers to put things into the cookie jar for our programs; for instance, if I'm writing a package of several programs, I can use a "Cookie" to notify each program that, for instance, certain tables have been set up.

The SST obviously needs a Cookie Jar as well, for we have many data structures operating. (For instance, the MMU tables are a multi-level data structure.)

After some thought, we decided not to use Atari's Cookie Jar. Why? Because the Atari Cookie Jar was already getting filled up with stuff! Why, there's a list of hundreds of "known cookies" identified in a recent magazine from Germany...

So, what we decided to do is set up our own, private place, called the Cockpit. (What else are you going to call the control area of an SST, I ask you?)

After running RAMnbmm.PRG, if you'll look in the Cookie Jar for an entry called "SR71" (after the SR-71A Blackbird that Sandy's Dad flew in), you'll find a pointer to the start of the Cockpit. Right now, the Cockpit is 40 entries long, formatted the same as the Cookie Jar.

Thus, we take up just one entry in the system Cookie Jar, and as many entries in our *own* Cockpit as we want, and still keep everything nice and easy to find.

We store two, sometimes three "Cookies" into this Jar; each point to something important. (The amount we store depends on how we handle disk drives; we have different methods needed to give us the flexibility to handle a wide variety of Atari Hard Drives and Floppies.)

Cookie Jar Pointer is always at $5A0; you can count on it. It always points to start of

Jar. If it isn't there, or is 0, there is a disaster of some form (or, you're running on an early TOS machine, which is *real* unlikely on an SST!)

$5A0 ————————> Cookie Jar Structure (presently $980)
$5A4: ————————> Physical TOP of SST RAM

Note: The SST RAM Initializer also sets location $5a4 to the top of SST RAM (either $0140 0000, for 4 meg, or $0180 0000, for 8 meg). This is really not that well documented, but is a *very* required operation if you want TOS to use your SST RAM. Also, we have been known to set the "top limit" slightly below the true physical limit, for example, 16K ($17F C000), to help stop "fencepost errors" in memory clearing routines; it's much better to lose a few bytes of RAM than it is to have a program crash because of a one-off error, which they do.

At system startup, many Cookies are placed into the Jar; they identify machine type, CPU, and so forth. Since a .Long (32-bit, 4 byte) entry can hold 4 letters, that's the length restriction on Cookie Jar names: 4 characters. It looks like this:

CookieName.Long, Cookie.Long
CookieName.Long, Cookie.Long
CookieName.Long, Cookie.Long
...for as many entries as are needed.

Final Entry:
0.Long, size-of-jar.Long

A 0.Long (4 bytes), followed by NumberOfEntries.Long,is the "Last Cookie". (To add an entry,you move the 0.Long down 8 bytes, and move the size-of-jar.Long down 8 bytes, and then add your entry where the old 0.Long and size-of-jar.Long were.

Note: the size-of-jar *DOES NOT EVER CHANGE*; you don't subtract one to indicate one less available entry! Should you run out of space, you are responsible for making a *new* cookie jar, larger than the old one, copying over all the old entries to it, and pointing $5A0.Long (Cookie Jar Pointer) to your jar - then ensuring your new Cookie Jar will stay resident in system RAM, which can *really* be a good time if you were not planning on writing terminate-and-stay-resident code...

When the program RAMnbmm.PRG is run, we add:

"SR71" ——> Pointer To SST Cockpit
"F117" ——> Pointer To Stealth Buffer

For TOS 2.06, we also add:

"_FRB" ——> Pointer to Disk Buffer

Cockpit Quick Reference

- "SR71" is a pointer to the starting address of the SST Cockpit, which is a data structure identical in shape to the ST Cookie Jar (e.g., Keep It Simple).

- "F117" is a pointer to a 32K-aligned, 64K long disk buffer. An F117 is also a Stealth Bomber, which is why we used the name "F117" for our "Stealth Buffer!"

Editor: Moan! I married someone capable of a pun like that?

...well, actually, okay, the F117 is a fighter; the B-2 is the "flying wing" Stealth

Bomber. We don't want to mess up anyone's security clearances - but would you believe someone in our family is working on the B-2? I guess it really is in the blood.

The "F117" and "_FRB" pointers just are absolute memory addresses of a disk buffer; there's no further indirection.

The "SR71" pointer points to the base of the SST cockpit, which shares the Atari memory structure; we figured that way, programmers wanting to access the Cockpit wouldn't have to rewrite their Cookie Jar access code (apart from the one indirection at start).

Caution: While accessing cookies in the cockpit, it is a violation of USAF and NASA rules to leave crumbs! They float around in zero-G conditions and get into instruments and controls, causing all sorts of problems.

$5A0 (Cookie Jar Pointer) —> Cookie Jar Table

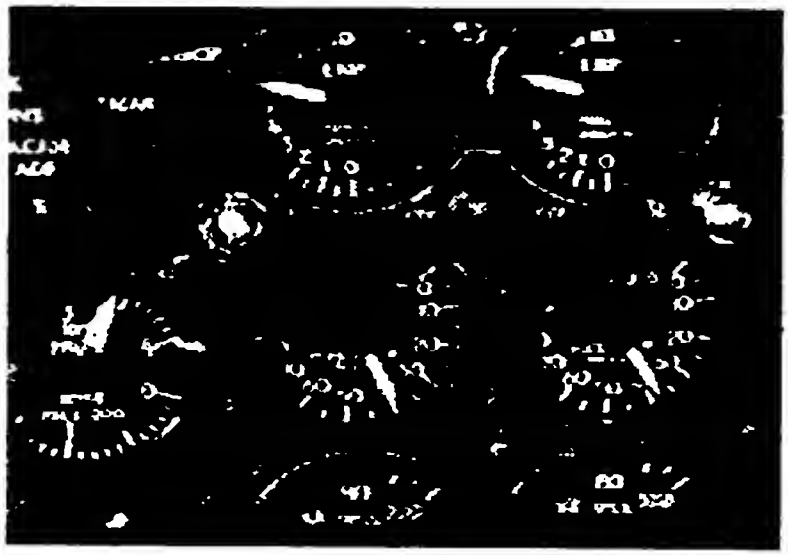
Inside the Cookie Jar table:

"SR71".ASCII.Long is followed by .Long pointer
to start of Cockpit. (Each entry is 8 bytes, remember).

(If there is no SR71 entry, RAMnbmm.PRG *HAS NOT BEEN RUN* or some disaster befell it.)

In the Cockpit, each entry looks just like the Atari Cookie Jar; the only difference I can think of is that the SR71 Cockpit is larger! (We initialize to 64 useable entries, Atari to 16.) Again, this is to let you easily re-use present-day Cookie Jar code.

There are a number of entries in the SR71 cockpit, roughly equivalent to the number of switches and dials in the SR-71A cockpit (grin). Some of them are there only for reference and debugging; for instance, I might be working on an MMU table and need to be able to quickly find its beginning, so I'll pop in a cockpit entry pointing to it. I tend to remove cockpit entries that are for short-term debugging. Others, though, are there for other programs to reference and modify.



Yes, this is an SR-71 Cockpit.

Of course, having *you* modify the cockpit entries very probably means "random death in the life support system" (a quote from Larry Niven and Jerry Pournelle). I am aware that some TT programs, for instance, Virtual Memory programs, are beginning to set up their own MMU structures; all I can say is, "beware". I am very open to any MMU programmers contacting me to set up a scheme where we can make our programs mutually aware of each other to prevent conflicts, from a simple "mutual exclusion" to a more complex memory management/partition scheme.

Finally, there *will* be additions and deletions to the cockpit entry list as the software evolves, so be careful with what I tell you about it here; it might change next disk. Don't write a program depending on them to stay the same from the first SST software version!

Revision 1.0 SST MMU Data Structure

You *will* need an MC68030 Manual's MMU section to make use of this; a manual on the 68851 PMMU can be helpful, too, as the 68030 MMU is very close to the 68851 (slightly less powerful). The 68851 manual has more introductory material and examples. Frankly, as someone who has learned something of the MMU, I would buy every book I could *find* about it, and get all the sample code I could; it gets rough.

Here are the *known* entries in the "SR71" Cockpit, as of this manual edition and before tonight's after-midnight hacking session:

"F117": The first entry is a repeat of the F117 Cookie Jar entry as a crosscheck on the software. (This helped me eliminate some subtle bugs in the memory management.)

The next entries tend to be related to the MMU (Memory Management Unit) tables. We use the MMU to decode memory into 32K blocks in two steps: first, decode to 16 megabyte "zones", then into 32K blocks. The first decode is essential to handling "24-bit" software that does not handles 32-bit addressing correctly; the second decode is for future software (okay, okay, I admit it; I've already written some of it!) for acceleration, compatibility, and whatnot. I needed to put "hooks" in here to keep the software options open for later.

"MTOP": points to the very top of the MMU table (the current translation table root pointer), where you point the MMU chip to via a MOVEC instruction. I added it purely for your reference; if you're just learning about the MMU, I thought you might appreciate a working example! You can then follow the pointers to see how a MMU table is set up, starting with "MTOP".

Note: MMU Entry "Pointers" are not "clean" on the lowest bytes; check out your MMU manual. The lowest bits are used as flags by the MMU. This is why the MMU entries do not start on "clean hex boundaries". Also, I tend to reserve a bit more memory for MMU tables than is strictly necessary; this is just style.

"M16A": 16 Megabyte decode table (256, $100, entries)
"MP00": 32K for 16 Megabyte decode table (512, $200 entries)
"MP01": 32K for 16 Megabyte decode table (512, $200 entries)
"MP02": 32K for 16 Megabyte decode table (512, $200 entries)

"M16A" is a pointer to the top of 16 megabyte MMU decode table. You'll find it repetitive (to say the least); nearly all of 256 short-form entries point to the "MP00" (lowest 16 Megabyte Zone, or Memory Page 00) early-termination table. In normal 32-bit mode, the only exception in the M16A table is the $01 entry (e.g., memory address $01xx xxxx, where x is anything), pointing to the table called "MP01"; this points to SST RAM. Even though SST RAM is presently limited to 8 megabytes ($0100 0000 - $017F FFFF), all 16 megabytes are fully decoded so I don't have to rewrite this code.

In "24-bit" mode, of course, *all* entries should point to the "MP00" table – except we do 24-bit at the moment using the MMU's Initial Shift = 8, which is cheap, tawdry, and works great, right out of the Spectre 3.0 code (I admit it). Only problem is switching back... which is why this table, and all that it implies, is available.

Note: Atari's 24BIT.PRG checks if it is running on a TT and does not work on the SST. Use our MMU24BIT.PRG instead.

"MP00" is a straight 1:1 decode, no address changing, for the $00xx xxxx address range of memory (0 to 16 megabytes). It uses short form entries.

"MP01" is a straight 1:1 decode, no address changing, for the $01xx xxxx address range of memory (16 to 32 megabytes). It uses short form entries.

"MP02" is a straight 1:1 decode, no address changing, for the $02xx xxxx address range of memory (32 to 48 megabytes). It uses short form entries.

Now, I wonder why we are decoding another 16 megabytes of memory? Now look. Are you the sort of person that reads the ending of a mystery novel first to find out what

happens? Of course not! Did you guess that Darth Vader was really Luke's father? Never! So wait for the next software version, ok?

Editor: "Sigh"... When he gets going like this, the biggest DELETE key in the world isn't big enough...

(Whether or not THIS MMU table scheme it is the most efficient I'm sure can be debated, but there are definite good reasons why it is set up the way it is for future software and hardware upgrades, which is a very, very broad hint. Fortunately, no one reads Appendixes. Seriously, please note that MMU table structuring for highest efficiency is a subset of computer science in general with a great deal of research, particularly since UNIX uses this so much, and fundamentally depends on the structure and flow of programs you are running.)

Entries MTOP, M16A, MP00, MP01, and MP02, as of this writing (might change tomorrow!) are located in SST RAM if at *all* possible. When RAMnbmm.PRG runs, it electrically switches *on* SST RAM, (starts it refreshing, tests it, zeroes it, and all that cool stuff), then MADDAlt's it so that TOS becomes aware that 4 or 8 megabytes of "Alternative RAM" (Atari's term) have been added. Before letting the desktop have a crack at SST RAM, though, RAMnbmm.PRG then promptly starts MxAlloc'ing (e.g., grabbing) room for the MMU tables, first-off. About 16K of memory is occupied by RAMnbmm.PRG for this process; this will show most clearly in the MEMSTAT6.PRG program, which shows used/free memory.

We put the MMU tables into SST RAM because the MMU hits on them a lot while decoding addresses, and everything stops while the MMU is doing that. If it were in ST RAM, everything would be slower; the 68030 has limited cache'ing of its MMU "Address Translation" entries, so you want MMU accesses to be real quick. This is yet-another-reason to run the SST with at least 4 Megs of SST RAM, even if it is not ultra-fast 60 nanosecond RAM; you'll really, really win in MMU table performance.

If there is no SST RAM, or RAMnbmm.PRG can't find it, which boils down to the same thing to an SST memory philosopher, then the program "drops back five and punts"; the tables are *very* regretfully put into ST RAM, and that's that.

"MMUT": This is not an address to an MMU table. Rather, it is an address into part of the MMU setup program buried inside of RAMnbMM.PRG. If you JSR to this address, the MMU tables will be completely reloaded from scratch (still at their old addresses, though.) You'd better JSR; we RTS if you enter here! This is for the rather obvious purpose of providing a "Get Out Of Jail FREE" card should you be experimenting with the SST's MMU and things go haywire; I always try to encourage people to "push the envelope", and this is a little help for you. In addition, if you follow the code beginning at MMUT's pointed-to address, you will see how a real, live MMU table is set up, which is a real handy example; for whatever reason, MMU books are really low on examples, which are the best way to learn. This is sort of one way of "passing forward" the help the computer community has given me.

Caution: Of course, whenever the MMU table goes sour, your machine is in fairly deep trouble. Also, the MMUT code can be tripped up by a variety of things that I can't do all that much about, since I can't anticipate what you'll do to the MMU tables. Still, it gives you SOME chance.

"MMUI": This is a pointer to another valuable portion of the MMU setup code. This subroutine simply wipes clean the address and data caches of the 68030, and PFLUSHA's (wipes clean) the address translation cache. It then returns to you; of course, the MMU will immediately begin reloading the ATC, and the caches, if on, will start doing their thing. Since it uses MOVEC instructions, you must JSR it in supervisor mode.

I provide this code entry, again, solely to help you experiment with the MMU. Programming the MMU can be extremely difficult to impossible if you do not make *CERTAIN* that the MMU is in a known state when you change it! This code sets the MMU into a known state. I strongly recommend you do what this code does before *and* after you change the MMU tables in nearly any way, and you may need to *SHUT OFF* the MMU

during the change too. (The symptoms of an unsecured MMU? Depending on just when the MMU does an ATC fetch, your program will run one time, then fail the next, for no discernible reason...)

This code cured probably among the most frustrating and time-consuming software bug I had during SST development, so I'm giving it to you.

"MBEG" & "MEND" are totally worthless debugging entries; they are there only to show me entry points in the code I used while developing it. I only noticed that they are still around when I displayed the Cockpit tonight to write this segment!

Other programs, which may or may not make it onto the first release of the SST disk (depending on what the Beta Testers have to say about them at our final session), use and sometimes *install* Cockpit entries, so you'll probably see more than I've mentioned here. I often use this as a way of notifying "the world" that a given significant program has run that has changed how the machine is going to respond. That's why I left a *lot* of room in the Cockpit.

Editor: GEARS? Oh, no...

I hope you've enjoyed this technical discussion, and we will now shift gears back to English.

Cockpit Data Structure Architecture

$5A0 —> Cookie Jar ($980 currently)

```
|   Cookie Jar Format:
|   Name.L, Pointer.L
    (Name.L, Pointer.L
|   ... (?? number prior entries)
|-------------------------------------------
|   "_FRB", Pointer.L—-> Points to 64K ST-RAM-Only Disk
                         Buffer, Currently NOT USED by SST

|   "F117", Pointer.L—-> Points to 64K ST-RAM-Only
                         Disk "Stealth Buffer"
                         Currently Used A Lot by SST

|   "SR71",Pointer.L—->  Aims At Cockpit Start: See Below

|   ... (any trailing entries)
|   0.Long, size-of-Jar.Long ($10 initially)
|-------------------------------------------
```

"SR71" Cockpit:

```
|       Cockpit Format:
|       Name.L, Pointer.L
|       ... (?? number prior entries)

|       "MTOP", Pointer.L ------> Start of MMU Tables
|       "M16A", Pointer.L ------> Start of 16 Meg Table
                                  Short-form, 16 Meg size
|       "MP00", Pointer.L ------> MMU Table, 0-16 Meg
|       "MP01", Pointer.L ------> MMU Table, 16-32 Meg
|       "MP02", Pointer.L ------> MMU Table, 32-48 Meg
                                  Short-form, 32K pagesize
|       "MMUT", Pointer.L -----> MMU Table Init JSR Entry
|       "MMUI", Pointer.L -----> MMU D/I/A Clear JSR Entry

|       ... (?? number more entries)
|       0.Long, size-of-Cockpit.Long ($40 initially)
```

The Index

_FRB 33, 35-37, 130-131, 134, vi
16K Cache 22, 24, 130, 133
24-bit Problem 28, 103, 113, 132
24BIT.PRG 103, 132
32-bit Clean 26, 84, 103, 113, 129-130, 132, i
454 Engine x-xv
6 Volt Battery 8
64 pin(s) 53-54, vi
64-pin Socket 59
64K Buffer 35-37, 130, 134
68000, Removing 45, 51, 53-56, 61, 66-67, 72, vi
68010 Microprocessor 25
68020 Microprocessor 25
68030, Installing 46-47, 57-58, vi
68030, Over-clocking 88
68030 Compatible 25, 29, 46, 68, 99, 109-110
68040 Microprocessor 25, 28
68881 FPU 28-29, 58, 117, 129, i, v, vii
68882 FPU 25, 28-29, 57-58, 117, 129, i, v-viii
7407 66, 119
7474 53-54
74LS04 121-122

A

Accelerators 1, 4, 11, 15-16, 21-22, 24, 26-27, 29, 45, 51, 97-98, 102, v, xiv
ACSI 35
Afterburners 25, 69, 72, 84-85, 87, 95, 105, xiv
AHDI 37, 46
Aircraft 81
Alligator-clip 8
Alpha Test 7, xvii
Aluminum Foil 8, 55-56
Angular Momentum 8
Antic Publishing 102
ASCII 13, 129, 131
Assembly Language 129
Atari 1040 ST 14, 30, vi
Atari 520 ST 13, 30, 51, 54, 67, 96, vi
Atari 800 6, 15, 40
Atari Dealer 52, 56, 67-68, 119, 121
Atari Developer 36-37, 98, xvi
Atari HDX 35-38, 46, 68, 90, vi
Atom 21, 39
Atomic Bomb 39, 41
AUTO Folder 34, 37-38, 45-49, 69, 81, 90, 104
AUTOMMUx.PRG 46-49, vi

B

B-2 Bomber 36, 130-131
Back to the Future ix
Backup(s) 7-9, 30, 46-47, 69-70, 96, iv
Badertscher, Ken 34
Baggage 62
Barb Hahn 1
Base-2 13-14, 82
Basement 1, 19, 62
Basic RAM Check 70
Battery 8, 52
Beard 21
Benchmark(s) 24, 27, 29, 87, 89
Berlin, Germany 32, 62
Beta Test 7, 37, 47, 108-111, 113, 134, v, vii, xiv, xvii
Beta Testers 37, 47, 108, 110-111, 113, 134, v, xiv, xvii
Binary 13-14, 82
Bit 1, 11, 14-18, 21, 25-26, 28-29, 35, 37, 42, 49-50, 52-54, 56-57, 63, 70, 72, 81-84, 92, 99-100, 103, 105, 108, 112, 119, 132, xii
Bit-flips 42, 50, 83
Black Company 75
Blackbird 1, 25, 95, 105, 129, 131, ii, xi, xiv-xv
Blank, George 6
BLITTER 29, 46, 53-55, 66, 69, 95, 104, vi
BLITTER Modification 46, 54, 66
Blood Sacrifices 58
BOINK Program 34, 71-73, 85, vi, xiv
Bomb(s) 2, 41, 70-71, 82-83, 89, 97-98, 103
Boot-sector 68
Booth, Mark xvii
Boston Rock Group 20, xiii
Bouncing-ball 71
Brain-dead 98
Brodie, Bob xvii
Buffer(s) 35-38, 119, 126-127, 130-131, 134, vi, viii
Bug(s) 25, 30, 40-41, 46, 49, 61, 77, 97, 103, 107-110, 115, 132, 134, vii, xvi-xvii
Bug-free 61
Bullets 56
Burst Mode 25-27, 29, 59, 70, 72, 81-82, 84-89, 96-98, 100, 111, i, vii-viii, xiv
Burst Mode Test 59, 86-87
Bus Bandwidth 89
Buzzwords 1-3, 17, 21, 70,

75
Byte 13-16, 22, 28, 70-71, 87, 129-130, v

C

Cache 22-25, 28-29, 56, 69, 72, 84-85, 95-100, 103-105, 110-111, 133, v, vii-viii
Cached Accelerator 22, v
CACHExxx.PRG 69
Camaro 85, ix-xv
Camera 19, 77
Capacitance 40-41
Capacitor 15-16, 42, 45, 55, 72-73, 121-122
Cat 6, 35, 49, 55, 62, 66, 78, 114, iv
Caution: 37-38, 55, 60, 70-71, 86, 89, 95, 100, 103, 110, 131, 133
CD-ROM 71
CBbit 64
Central Processing Unit 16-18, 21-22, 25-26, 30, 50, 65, 68, 71, 79, 83-85, 88-89, 98, 104, 110, 117, 130, v, vii-viii
CHANGE Button 101
CHANGE Column 101
Chevrolet xi, xiii, xv
Children 1, 19, 31-32, 51, 62-64, 76, 78-79, 91, 93, 115, ix, xii, xiv-xv, xvii
Chromax 28
Clear Flags 99, 101, vii
Clear Memory Flag 99, vii
Clear Memory 48-49, 99, 101, 130, 133, vii
Clear Screen 23, 50
Clemans, Samuel 115
Clever-fast-copy-loop 36
Clippers 46, 54
Clock Speed 21, 27, xiii
Cockpit Data Structure 65, 129-134, vii
Codehead Software 26, 100
Coffee 5, 18
Cold Solder Joint 18, 36, 40, 42, 46, 48-55, 58, 66, 71-72, 114, 119-122, x, xii, xvii
Cold Start 48-50, 71, x
COLDBOOT.PRG 46, 48-49, 71, vi
Color Video 3, 14, 28, 79, 102, 107, 110
COMDEX 64
CompuServe 116
CompuServe ID 116
Computer Show Exhibitor 63-64
Computing Capacity 4
Conductive 50, 55-57, 93
Conductive Foam 55-57
Configuring the SST 45, 81, 84, xv
Consultant 3
Contact-enhancers 67
Contiguous Memory 97
Continuity 54, 121
Control Panel Extension 100, 104
Control-Alt-Delete 48
Cook, Glen 75
Cookie Jar 35, 37, 129-132, 134
Cookiename 130
Cookies 129-131
Coprocessor 25, 28-29, 57-58, 117, 129, i, v-viii
Copy Loop 27, 36
Cosmic Rays 42-43
CPU 16-18, 21-22, 25-26, 30, 50, 65, 68, 71, 79, 83-85, 88-89, 98, 104, 110, 117, 130, v, vii-viii
CPU Cycle 17
CPX 100, 104
Cray, Seymour 40, 43, 76
Creative Computing Magazine 6, 10
Crud 66-67, 72, 76
Current Notes Magazine 20, 25, 93, 115

D

Dead Man's Land 40, 83
Dead ST 66
Dead-on-arrival 63
Dealer 52, 56, 67-68, 119, 121, 123
Debug 12, 20, 26, 40, 44, 77, 86, 109, 131, 134
Decimal 82
Decode Memory 132
Dedication Page xii
Delay Loop 14-15, 17, 23, 27, 30, 37, 40-41, 59, 68, 71, 85-86, 88, 97-99, 102-103, 107, 109-111, 113, 127, vii-viii
Demonstration 13, 31, 61, 63-64, 92
Desk Accessory 26, 46, 90-91, 104
DESKTOP.INF 46, 53, 96
Desktop Information 66, 96, 103
Desolder 51, 53-54
Detent 25, 95
Diet Pepsi 5
DIP 58
Disclaimer ii-v
Disk Directory 12, 37, 69, 81, 101
Disk Drive 7-11, 14, 26-27, 30, 33-38, 44-47, 49-50, 52-55, 59-63, 65-73, 75, 78, 87, 89-90, 96, 101-102, 104, 107, 109-112, 114, 119-122, 129, v-xi, xiii-xv, xvii
Disk Driver 14, 33-38, 46, 60, 68, 90, 96, 129, vi
Disk Format 102, 112
Disk Handler 14, 33-38, 46, 60, 68, 90, 96, 129, vi
DISKxx.PRG 35-38, 60, 71-72, 90
Disney, Walt 78
DMA 16, 27, 66-67, 102, 119-120, 127, v, viii
Doc from Oz 32
Dot 2-5, 11-14, 18, 23, 50, 55, 58, 76-77, 121, v, viii
Double-click 12, 34, 47, 70, 90, 98, 103-104, 107
Double-sided 46, xv
DRAM Controller 89, 118, xiv
DRAM 15-16, 21-22, 30, 50, 59, 70, 89, 118, v, xiv
Dream 1, 8, 20, 79, 93, 106, 123, ix, xi, xv
Dungeon Master 13
DynaCadd 28
Dynamic RAM Controller 89, 118, xiv
Dynamic RAM 15-16, 21-22, 30, 50, 59, 70, 89, 118, v, xiv

E

Early-termination 132
Editor: 2, 4, 6, 14-15, 18-19, 21, 25, 30-31, 33-34, 36, 40, 42-43, 48-49, 56, 59, 62-63, 65-66, 68-73, 81-82, 84-87, 89, 95-97, 99, 102-103, 105-107, 113-115, 129-130, 133-134, xvi-xvii
Edwards AFB x
Einstein, Albert 39, 43, 91
Ejection Seat 65, 73, ii, xi
Electric Potential 50
Email 116
Emulator 6, 28, 45, 61, 64,

xvii
Engine 22, 24-25, 45-60, 69, 87, 89, ii, v-vi, x-xiv
Engineer 11, 16, 20-21, 25, 39-40, 43, 50, 69, 113, xiv-xv
Entropy 8
EPROM 30, 57
Error 8, 34, 49, 77, 82, 98, 103, 108, 111, 130, xiii, xvii
Ever-so-common 39
Example 2, 26, 39-41, 46, 49, 72, 76, 81, 86, 101-103, 107-108, 110, 115, 130-133
Exception 37, 62, 98-99, 109, 132, ix
Exhibitor 63-64
Expansion Connector 28, 125-128, i, v, vii-viii

F

F-117 Stealth Fighter 35-36, 130, vi
F117 Buffer 36-38, 130-132, 134, vi
Fang 6, 63-64, 78, iv, xviii
Fast enough 18, 105
Fast-copy 27, 35, 102
fastRAM Initializer 37, 69-72, 81, 85, 88, 90, 96, 104, 130, vii
Fax 0-1, 68, 109, 115, 118, 123-124, xvi
FBOINK Program 71-73, 85
Fetch 12, 17, 21, 23, 26, 50, 82-86, 88-89, 133-134, xiii
Fire-breathing x-xi, xiv
Flags 37, 99-103, 110, 132, vii
Flashlight 59
Flatblade 55-56
Floating Point Unit 25, 28-29, 117, 129, vii
Foil 55-56
FOLDRxxx 69
Foo 13
Foreigners 31
FPU 25, 28-29, 117, 129, vii
Fred 9
Furb 33, 35-37, 130-131, 134, vi

G

Gadgets by Small, Inc. 0-1, 10, 55, 61, 63, 89, 108, 115-116, 119, 121, 123, ii-iv, vii, xv-xviii
Gamma Test xvii
GCR 6-8, 10, 20, 28, 41, 78, 102-103, 107-113, 115, 121-122, 132, vii-viii, xiv, xvi-xvii
Gearhead 69, 134, x-xi, xiii
GEM 38, 47, 104
GEnie 116, xvii
GEnie ID 116
Germany 31-32, 62, 124, 129
GFA Basic 103
Gibson, Mel 87
Gigabyte 103
Glitch Chip Corporation 41
Glitch 41, 47-48
GLUE Chip 18, 22, 66-67, 89
Grab 12, 17, 21, 23, 26, 50, 82-86, 88-89, 133-134, xiii
Gravity xiii
Greenblatt, Jeff xvii
Guru 56, x

H

H-bomb 41
Hacker 3, 10, 20, 61, 63, 71, 76, 105, 129, x
Hackercon 71
Half-moon 54-55, 121
Hamacher-Gatzweiler, Karl 124
Hard Disk Driver 14, 33-38, 46, 60, 68, 90, 96, 129, vi
Hard Disk Modification 46, 66, 102, 110, 119-120, vii
Hardware Options 47, 56, 86, 90, 100, 109, 116-118, vi-vii
HDX 35-38, 68, 90, vi
Heartbeat 21, 58
Heat Failure 30, 71
Heat 30, 43, 54, 71-72, 121
Heat-Sink 52
Heidlebaugh, Gary 95, ii
Heisenberg Uncertainty Principle 34, 39-44, 109, vi
Hero 91-92, xvii
Hertz 16
Hexadecimal 82, 132
HG Computers 124
High Flight 106
High Speed 26, 28, 35, 40-41, 49, 69-70, 83, 85, 88, 97-100, 102-104, 117, 119, vii, xiii, xv
Hip-high Cast 32
Hold-downs 56-57, 67
Holmes, Sherlock 40
Horizontal Blank 3, viii
Hotz, Jimmy 10
Hudson, Tom & Elizabeth 10
Human Being xv-xvi

I

IBM 24-26, 28, 30, 39, 42, 56, 79, 117
ICD 35, 38, 46, 68, vi, xvii
Icon(s) 4, 12, 47, 69, 95-96, 99, 104
Illudium Phosphate 21
In-circuit-emulator 54
Incompatible Software 99-100
Inductance 41
Industry-wide 13
Initialise SST RAM 37, 69-72, 81, 85, 88, 90, 96, 104, 130, vii
Insulator 59
Intelligent Loop 97-98, viii
Intensity 5, 11
Intercept Disk Access 36
Interesting Times 64
Internet 116

J

Jaggies 4
Jamaica 1
Jargon 1-3, 17, 21, 70, 75
Johnson, Kelly ii, xv
Jolt Cola 16
JP-6 Fuel 65, i, xi, xiv

K

Karma Points 46, 83
Keypress 7, 48-49, 71, 96
Kids 1, 19, 31-32, 51, 62-64, 76, 78-79, 91, 93, 115, ix, xii, xiv-xv, xvii
Kilobyte 13, 64
Kitty Litter 62

L

LaserWriter 78
Last-access 23
Layout 26, 30, 89, 109
Leaky DRAMs 16, 48, 59
Lear Jet 61
Legendary xi
Limiting Factor 15
Loop(s) 23-24, 27, 29, 37, 59, 71, 85-86, 97-99, 103,

107, 110, 113, viii
Low Speed 87, 98-99, vii
LS-6 Engine xi-xiii, xv

M

M16A 132-134
Mach 1, 25, 70, 95, 105, v, xi
Machined-pin Socket 51, 54-55
Macintosh IIfx 15, 78, xv
Macintosh 6, 14-15, 25-26, 28, 42, 61, 64, 78, 101-103, 105, 107-112, 115, 117, 122, xiv-xvii
Maddalt 37, 104, 133
Magee, John Gillespie 106
Malloc 100, 102-103, 115
MBEG 134
MCU 17-18, 22, 59, 67
Medium Speed 70, 98-99, 103, vii
Mega 12 27, 96, xv
Mega 4 13-14, 96, xiv
Mega ST 13-15, 26-30, 33, 45, 50-56, 59, 62, 67-68, 72-73, 96, 119, 123, 127, iv, vi, viii, xiv-xv
Mega-hurts 16
Megacycle 16
MegaHertz 16-17
MegaTalk 6, 20, 65, xiv-xv
Memory Address 131-133
Memory Allocation 100, 102-103, 115, 133
Memory Control Unit 17-18, 22, 59, 67
Memory Cycles 17-18, 22
Memory Management Unit 16, 18, 25, 66, 103, 107, 110, 129, 131-134, v, vii
Memory Request Flag 100, 102, vii
Memory Request 26, 100, 102-103, vii
MEMSTATx.PRG 100, 104, 133, vii
Menu 10, 46, 69, 72, 78, 84, 95, 98, 101, 111
Merlin Group 6, 20, 60, 62, 66, 92, 119-120, 123, i, vii
MFP Chip 97
Microprocessor 16-17, 117-118, 129, vii
MIPS 27, 29, 87, 89
Miracle 2, 39, 86, ix
Mission Impossible 42
MMU 16, 18, 25, 66, 103, 107, 110, 129, 131-134, v, vii
MMU Table 107, 110, 131-134
MMU24BIT.PRG 103, 113, 132
MMU1 133-134
MMUT 133-134
Modus Operandi 63
Monday 56, 59, 62, 116
Moniterm 59
Monitor Display 4, 11-12, 27, 75
Monitor 2-5, 11-12, 15, 17-18, 22, 27, 58, 61, 65, 75, 79, 107, 110, 112, viii
Monochrome 2-3, 14, 28, 65, 109-110
Moore, Dan 10, 78
Motherboard, ST 30, 51-53, 66, 73, 119-120, 127
Motorola 25, 28, 117, 127, xiv
Mouse 61, 78, 87, 101, xiv
MP00 132-134
MP01 132-134
MP02 132-134
MTOP 132-134
MultiFinder 115
Mumbo-Jumbo 69
Muzak 82
Mxalloc 133

N

Nanosecond 39-43, 57, 82-83, 88, 117-118, 133
Needle-nosed Pliers 54, 56
Network 6, 116, xv-xvii
NEWDESK.INF 96
Newsletter 38, 107-108, 115, xvi
Nintendo 77
Nippers 51
Niven, Larry 131
Normal Wait States 82, 84-86, 88-89, vii-viii
Note: 15, 27, 46, 48, 52, 55-56, 67, 71, 76, 83-84, 101, 111, 116-117, 125-126, 130, 132-133, xii
Nybble Mode SIMMs 26, 29
Nybble 13, 26, 129

O

One-kay 13
Online 6, 116, xv-xvii
Operating System 102
Optimizing Your SST 58-59, 70, 72-73, 81, 83, 86, 88, 96, 110, vii
Optional SST Hardware 47, 56, 86, 90, 100, 109, 116-118, vi-vii
Oscillator Frequency 81, 92, 127
Oscillator Installation 58, 70, vi
Oscilloscope 8, 40-42
Other Editor 3, 5, 13, 16, 26, 30, 33, 35-37, 40, 49, 62, 65, 69-71, 73, 85, 87, 97, 113, xiv, xvii-xviii
Over-clock 88
Over-run RAM 82, 84
Overheat 55, 59
Oxidation 66-67, 72, x

P

Page Mode SIMMs 26, 57, 83, 117-118
PartCopy 63
PEEK 12, 102-103
Pepsi 5, 18
Persistence 48, 70, 89
Phone 1, 32, 68, 109, 115-116, 118, 123-124, xvi
Phosphor 2-5
Pin 1 Orientation 54-55, 57-58, 121-122, viii
Pipeliner, 68030 25, 85
Pixel 3, 14, 18, 78
PLCC Chip 67
Pliers 46, 51, 54, 56
PMMU 25, 131
Pocket-protectors 129
POKE 12, xii
Pournelle, Jerry & Roberta 20, 131
Power Supply 30, 51-53, 55, 57, 59-60, 63, 65-66, 72-73, 89, 119, vi
Power Without The Price 25-26, 28, xv
PRGFLAGS.PRG 99-103, 105, vii-viii
Processor-direct 28
Propagation 40
Pull-down Menu 101
Pyewackette 6, 35, 49, 55, 62, 66, 78, 114, iv

Q

Q-index 104
Quick Hard Disk Fix 66, viii

QuickST 29

R

Radio Shack 51, 121
RAM Cycles 17-18, 22
RAM 11-18, 21-30, 33-39, 41-43, 45-50, 53, 56-60, 65-66, 68-72, 75, 79, 81-90, 96-105, 108-111, 113, 117-118, 130-133, i, v-viii, xiv-xv
Ram33mm 72, 85-86
RAMDISK 26, 49-50, 70, 79, 81, 104, vi
RAMnbmm.PRG 33, 37-38, 47, 49, 70-72, 81, 85-86, 88, 90, 104, 129-131, 133, vii
Random Access Memory 15
Raving Idiot Part Supply 41
Read Only Memory 38, 53, 56, 59, 65-68, 102, 111, vi
Reassemble your Mega 55, 73, vi
Reboot 61
Red Car 85, ix-xv
Refresh Test 70
Registration Card 107, 114-115
Reseating Chips 67
RESET Button 48-50, 53, 61
RESET 33, 48-50, 53, 61, 66, 68, 70-71, 85-86, 96, 119, 126
Reset-proof 49-50, vi
RGB 11, 14, 28
Richardson, George 6, 20, 60, 62, 66, 92, 119-120, 123, i, vii
Ringing 1, 41-42, 89
Rock Bottom Price 16
Rockman 20, xiii
Rogovin, Bruce xvii
ROM 38, 53, 56, 59, 65-68, 102, 111, vi
Rust Never Sleeps 4

S

Scan Line 2-4, 14-15, viii
Scholz, Tom 20, xiii
SCSI 14, 109-110, xiv
Serviceperson 67
Sharp-nosed Clippers 46
SHIFTER 11-18, 21-22, 50, 66-67, v
Shotgunned RAM 50
Show Survival Kit 7, 63
Shwarzenneger 9
SIMMs, 1x8 57, 66, 83, 117-118
SIMMs, 1x9 56, 66, 117
SIMMs, 4 Megabyte 57, 83, viii
SIMMs, Installing 45, 56-57, 71, 117, vi
SIMMs, Removing 57, 66
Size-of-cockpit 134
Size-of-jar 130, 134
Sleeping Beauty 58
Small, David 3, 5-6, 10, 13, 16-17, 21, 25-26, 30, 33, 35-37, 40, 48-49, 61-62, 65, 69-71, 73, 77, 84-85, 87, 95, 97, 113, 123, i-ii, v, xi, xiv, xvi-xviii
Small, Eric 6, 10, 19, 62, 77-79, 91-94, vii, ix, xii
Small, Jamie 6, 10, 62, 78-79, 115-116, ix, xii, xv
Small, Jenny 6, 10, 19, 62, 75-80, vi, ix, xii
Small, Sandy 2, 4, 6, 10, 14-15, 18-19, 21, 25, 30-34, 36, 40, 42-43, 48-49, 56, 59, 62-63, 65-66, 68-73, 78, 81-82, 84-87, 89, 95-97, 99, 102-103, 105-107, 113-115, 129-130, 133-134, i-ii, ix-xi, xiii, xvi-xvii
Snowdog 76-77
Snowman 75-80, vi
Socketed 68000 45, 51, 53, 55, 61, 72, vi
Solder 40, 42, 46, 51-55, 58, 66, 72, 119-122
Solder-mask 54
Solder-pad(s) 51, 54
Solder-sucker 54
Solder-tail Socket 55
Soldering Iron 46, 50-52, 54
Sonic Boom 95, xi
Spectre 128 6-7, 107, 113, xii, xvi
Spectre GCR 6-8, 10, 20, 28, 41, 78, 102-103, 107-113, 115, 121-122, 132, vii-viii, xiv, xvi-xvii
Speed-of-light 89
SR-71 1, 25, 95, 105, 129, 131, ii, xi, xiv-xv
SR71 Cockpit 65, 129-134, vii
SST Configuration 45, 81, 84, xv
SST Diagnostics 30, 45, 48, 56, 59-60, 65-67, 70-73, 84, 96, 99-100, 103-104, 111, 132, vi
SST Fine-tuning 45
SST Flexibility 81, 90, vii
SST RAM Compatible 28-30, 33, 46, 97, 99-100, 105, 107, 113, vi-vii
SST RAM Flag 99-100, vii
SST RAM Initialization 37, 69-72, 81, 85, 88, 90, 96, 104, 130, 134, vii
SST RAM 0-1, 4, 6, 10, 12-17, 20-30, 33-60, 63, 65-73, 75-77, 79, 81-90, 95-119, 123-134, i-viii, xiv-xv, xvii-xviii
SST RAM Test 27, 34, 42, 45-47, 49, 56, 69-73, 81, 83-85, 87-89, 99-100, 104-105, 114, 133-134, v-vi, xiv, xvii
SST Release Disk 34, 36, 46, 68-71, 103-104, 107
ST RAM Flag 100, vii
Stack-overflow 63
Stallone 9
Star Trek 69
Star Wars 65
Static Cache 22
Static Electricity 50-51, 55, 66-67, 114, vi
Static Precautions 15-16, 22, 26, 50-51, 55-58, 65-67, 83, 114, 121, v-vi
Static Precautions 51, 56-58, 65, vi
Static RAM 15-16, 22, 26, 56, v
Static-proof 51
Stealth Bomber 36, 130
Stealth Fighter 36
SUPEDIT 7, 63
Supra 35, 38, 46, 68, vi
Swiss Army Knife 31
Syquest 7, 9, 62

T

Technical Note: 34, 37, 86, 96, 129
Techno-dweeb xvi
Technodetails 99
Technoengineer 35
Technology 25, 43, xiv
Technotalk 103, vii
Technowhiz 3
Terminator 10, 40
Tesla, Nikola 20, 91-94
Throttle 25, 73, 87, 95, 97, xiii
Tick 82-83, 87-88

Timer 97-98
Titanium xi
Toad Computers 123
Toolkit 51, 62
TOS 2.05 29-30, 34, 37, 68-69, 100, i, vi
TOS 2.06 1, 29-30, 33-34, 37, 65, 68-69, 95, 97, 99-100, 113, 130, i, vi
TOS ROMs 30, 38, 53, 56-57, 59, 65-67, vi
Troubleshooting 34, 42, 65, vi
TSTnbmm.PRG 70, 72, 81, 84
TSTnbmmD.PRG 85-86
TT Architecture 96
TT Compatible 28-29, 46, 68, 97, 109, v
TT RAM 34-35, 46, 96, 100, 104, 109, 111, 113
Tuning Wait States 84-86, 88, vii
Tweek 67
Twister 14, 27

U

Undiscovered Country 105, ii
Unix 25, 75-79, 133
User Groups 3, 26, 50, 100, 109, 115-116, xvi-xvii
User-installable 47

V

Vader, Darth 133
VBL 3, 109
Vertical Blank 3, 109, viii
Video Contention 1, 21, 24, 26, 29, 95, 98-99
Video Cycle 17
Video Display 2-4, 11-12, 14, 16, 21, 27, 41, 63, 75, 77-78, 86, 96, 100-102, 134, viii
Video Memory 11-14, 16-18, 21-23, 25-29, 33, 50, 65, 71, 96, 100, 102, v, viii
Video Refresh 3, 16, 18, 50, v

W

Wait States 69, 81-89, vi-viii
Warm Start 48-49
Warning 9, 27, 52
Warranty 52, 107, ii-v, xvii
Wheeler, Doug 20, xvii
Wire-cutters 51
Wizards xv
Woof 3, 13

X

X-acto 46, 54, 67, 119-120

Z

Zerostore 103, 108

feel the need... THE NEED FOR SPEED

SR-71A Blackbird

Speed - Classified (Mach 3+)
Engines - Pratt & Whitney JT-11D Turbofans
32,000 lbs. thrust w/ afterburners
Ceiling - Classified (80,000+ ft.)
Fuel - JP-6

the world's fastest plane...

SST

...the world's fastest Atari

Introducing the fastest, most compatible, most flexible accelerator available for the Atari Mega ST: the splendiferous totally cool ***68030 SST****. With its screamingly fast 68030 microprocessor, up to 8 meg of fastRAM (to 12 meg total), asynchronous design, and special compatibility software by Dave Small, the SST is any "ST pilot's" dream...*

...The dream can be ***yours****! So bring your Mega ST into the 90's, and turn it into one of the fastest machines on the planet!*

From the people who brought you Macintosh emulation on the Atari (and whose Dad actually flew the SR-71).

68030 SST

Speed - 18 to 40 MHz asynchronous design; up to 12 times faster than a Mega ST
Engine - MC68030 Microprocessor with Caches and Burst Mode Afterburner
Ceiling - up to 12 Megabytes of RAM
Fuel - up to 8 Megabytes of 32 bit fastRAM in tandem with a 20 or 33 MHz oscillator
Operating System - TOS 2.06 (Mega ST^4), with the new Atari desktop
Expansion - processor direct slot; Chromax Super Video board already "in rollout"
User Upgradeable - when ever you want to

SST prices start at $599.00

Gadgets by Small SSTSSTSSTSST

40 W. Littleton Blvd.; #210-211 • Littleton, CO 80120 • (303) 791-6098 • Fax (303) 791-0253